VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ PRO ROZPOZNÁNÍ A KONTROLU OBJEDNÁVEK SPEDICE

A TOOL FOR RECOGNITION AND VERIFICATION OF SPEDITION ORDERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VOJTĚCH KALIVODA**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Kalivoda Vojtěch**
Program: Informační technologie
Název: **Nástroj pro rozpoznání a kontrolu objednávek spedice**
A Tool for Recognition and Verification of Spedition Orders
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se s problematikou vývoje webových aplikací s netriviálním backendem pro zpracování dokumentů.
2. Analyzujte a popište problematiku zpracování objednávek spedice.
3. Navrhněte architekturu systému pro rozpoznání a kontrolu objednávek.
4. Shromážděte dostatečnou datovou sadu pro vývoj a ověřování vyvíjeného systému.
5. Prototypujte jednotlivé části systému a ověřujte jejich funkčnost na datové sadě.
6. Navrhněte uživatelské rozhraní vytvářeného systému.
7. Implementujte webovou aplikaci pro rozpoznání a kontrolu objednávek spedice.
8. Testujte systém s uživateli na reálných datech a iterativně jej vylepšujte.
9. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 až 6.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem této práce je návrh a implementace webového nástroje, který usnadní práci dispečerům spedičních a dopravních firem pomocí automatizovaného rozpoznání důležitých informací v objednávkách. Díky rozpoznání nemusí být všechny informace ručně přepisovány dispečery, což vede k ušetření času. Rozpoznávání objednávek je postaveno na vyhledání entit v dokumentu, reprezentace jejich okolí vektory za pomocí word2vec modelů a následné klasifikace pomocí konvolučních neuronových sítí. Nástroj dokáže v reálném čase rozpoznat 20 typů informací s průměrnou úspěšností 72.35 %. V rámci práce byl shromážděn dataset necelých 1 700 objednávek a 141 z nich bylo anotováno. Součástí práce je webová aplikace, která slouží jako rozhraní pro nástroj a sběr dat.

Abstract

The aim of this work is to design and implement a web tool that will facilitate the work of dispatchers of forwarding and transport companies through automated recognition of important information in orders. Thanks to the recognition, not all information has to be manually rewritten by dispatchers, which saves time. Order recognition is based on finding entities in a document, representing its surroundings with vectors using word2vec models and subsequent classification using convolutional neural networks. The tool can recognize 20 types of information in real time with an average success rate of 72.35 %. As part of the work, a dataset of almost 1 700 orders was collected and 141 of them were annotated. Part of the work is a web application that serves as an interface for the tool and data collection.

Klíčová slova

zpracování dokumentů, rozpoznání objednávek, extrakce informací, word2vec, konvoluční neuronové sítě, webová aplikace

Keywords

document processing, order recognition, information extraction, word2vec, convolutional neural networks, web application

Citace

KALIVODA, Vojtěch. *Nástroj pro rozpoznání a kontrolu objednávek spedice*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Nástroj pro rozpoznání a kontrolu objednávek spedice

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Kalivoda
9. května 2021

Poděkování

Rád bych poděkoval svému vedoucímu panu prof. Ing. Adamu Heroutovi, Ph.D. za odborné vedení a užitečné připomínky.

Obsah

1 Úvod **3**

2 Problematika zpracování dokumentů **4**
2.1 Zpracování přepravních dokumentů 4
2.2 Extrakce informací z dokumentů 5
2.3 Použitý dataset 8

3 Použité nástroje a znalosti **10**
3.1 Použité nástroje 10
3.2 Reprezentace slov 11
3.3 Klasifikační metody a modely 14

4 Návrh řešení **21**
4.1 Rozpoznávání objednávek 21
4.2 Word2vec modely 25
4.3 Klasifikační modely 27
4.4 Proces anotace 29
4.5 Proces testování 30

5 Návrh webové aplikace **32**
5.1 Analýza požadavků 32
5.2 První návrh 33
5.3 Druhý návrh 34
5.4 Finální návrh 34

6 Implementace **36**
6.1 Datové složky 36
6.2 Rozpoznávací knihovna 36
6.3 Klientská část aplikace 38
6.4 Serverová část aplikace 39

7 Hodnocení rozpoznávacího systému **40**
7.1 Dosažená přesnost klasifikace 40
7.2 Porovnání s konkurencí 40
7.3 Rychlost rozpoznávání 43
7.4 Celkové zhodnocení 44

8 Závěr **45**

Literatura **46**

A Plakát **48**

B JSON schéma anotace **49**

Kapitola 1

Úvod

Tato práce se zabývá zpracováním a extrakcí informací z objednávek na přepravu. Výstupem práce je produkt, který rozpozná důležité informace v objednávkách a převede tyto informace do strojově čitelného formátu.

Spoluzaložil jsem technologický startup Cargotic, zabývající se digitalizací a automatizací v oboru logistiky. Momentálně poskytujeme speditérským a dopravním firmám administrační systém pro dispečery a mobilní aplikaci pro řidiče. Spousta především malých a středních speditérských a dopravních firem využívá pro správu nad svými přepravami papírové šanony nebo excel soubory. V lepším případě pak jednoduché administrační systémy, které požadují jen základní informace bez výraznějších validací. To ale způsobuje velmi omezenou funkcionalitu bez možnosti rozšířenější práce s daty nebo automatizace. Dispečeři jsou pak často zvyklí překopírovávat data z objednávky bez jakékoli validace či uspořádání. Pro firmy, které chtějí mít větší kontrolu a přehled nad daty je pak těžké přecházet na systém, který po jejich lidech vyžaduje preciznější zadávání informací, které je i více časově náročné. Naši platformu využívá na každodenní bázi několik společností, a právě zdlouhavé zadávání informací je jeden z nejčastějších problémů, se kterými firmy, respektive jejich zaměstnanci bojují.

Právě zrychlení zadávání a tím pádem i ulehčení práce by měl vyřešit produkt této práce. Produkt bude webová aplikace, později implementovaná do administračního systému Cargotic. Aplikace bude na vstupu očekávat objednávky na přepravu, které budou zpracovány a rozpoznané pole automaticky vyplní formulář, který by jinak dispečer spediční nebo dopravní firmy vyplňoval ručně. S aplikací budou pracovat i méně technicky zdatní uživatelé musí být tedy dostatečně intuitivní a uživatelsky přívětivá. Aplikace by také měla rozpoznané informace náležitě zobrazit, nejlépe jako výstřižky ze samotného dokumentu s vyznačenými hodnotami, které systém nalezl a doplnil. Informace zobrazené přímo ve webové aplikaci proces kontroly nejen zrychlí, ale také minimalizují možné chyby při kontrole.

Samotné rozpoznávání je postaveno na nalezení entit v dokumentu a jejich následné klasifikaci pomocí konvolučních neuronových sítích. Klasifikace jednotlivých polí v dokumentu probíhá na základě pozice a sousedních prvků. Texty sousedních prvků jsou reprezentovány vektory, které jsou generovány pomocí vlastních word2vec modelů.

Následující text podrobněji specifikuje problém a popisuje teorii použitou v návrhu rozpoznávání. Dále je popsán samotný návrh rozpoznávacího modulu, uživatelského rozhraní a API serverové části. Na návrh navazuje implementace, která popisuje rozdělení a jednotlivé části celého produktu. Na závěr práce jsou shrnuty dosažené výsledky a možné rozšíření do budoucna.

Kapitola 2

Problematika zpracování dokumentů

2.1 Zpracování přepravních dokumentů

V dnešní době se přenáší obrovské množství informací v dokumentech, které jsou pro firmy napříč nejrůznějšími obory nepostradatelné. Dokumenty nesoucí takové informace mohou být **strukturované**, pokud je jejich šablona fixní a informace v dokumentu se nachází vždy na stejných pozicích. Mezi takové dokumenty může patřit například daňové přiznání. Dalším typem dokumentů jsou **částečně strukturované**, to mohou nejrůznější účtenky, faktury a právě i objednávky na přepravu. Tyto dokumenty mají společné prvky, často se v nich nachází stejné či podobné druhy informací, ale jejich šablona a tvar se může měnit v rámci firem nebo i poboček. Posledním typem jsou dokumenty **nestrukturované**, takové dokumenty často obsahují více přirozeného textu, jedná se o různé články nebo dopisy.

Rychlé zpracování takových dokumentů je pro mnoho společností klíčové. Dokumenty jsou často zpracovávány ručně a systém, který dokáže automaticky rozeznat důležitá data v dokumentech má pro firmy velkou hodnotu v ušetření a zefektivnění práce zaměstnanců.

Objednávky na přepravu patří mezi částečné strukturované dokumenty, a proto se dále v této práci bude zabývat tímto typem dokumentů. Objednávky se objevují v nejrůznějších formách, často vícestránkové, někdy více uspořádané do tabulek a formulářů, jindy skládající se více z přirozeného textu. Tyto vlastnosti vytváří poměrně složitý problém, pro jehož vyřešení je potřeba kombinace technik z NLP[1] a počítačového vidění.

Základem pro úspěšné rozpoznání a zpracování objednávek je správné přečtení těchto dokumentů. Objednávky se v drtivé většině případů vyskytují ve formátu pdf. Dokumenty ve formátu pdf mohou být digitálně vytvořeny nebo zpracovány tak, že je v nich možné hledat a číst z nich potřebné informace jako text nebo pozice. Určité množství objednávek jsou pak oskenované a vyfocené objednávky s různou kvalitou. V takovém případě není možné texty a jejich pozice jednoduše přečíst, ale je nutné vyextrahovat potřebné informace pomocí OCR[2].

Pro zvýšení přesnosti při rozpoznávání informací je nutné mít dobrý přehled o 2D rozložení dokumentu, proto se při čtení dokumentu, a to i v případě použití OCR, získává nejen samotný text, ale i pozice textů. Texty v dokumentech jsou rozděleny do textových elementů. Každý textový element obsahuje plochu, na které se nachází a jeho textový obsah,

[1] NLP – Natural Language Processing – zpracování přirozeného jazyka
[2] OCR – Optical Character Recognition – optické rozpoznávání znaků

což může být slovo, věta nebo řádek textu. Klasifikace elementů do jednotlivých kategorií pak probíhá nejčastěji pomocí ML[3] modelů nebo pomocí předepsaných pravidel.

2.2 Extrakce informací z dokumentů

Vedle pokusů o komerční řešení dále popsané v sekci 2.2.4 existují také vědecké články zabývající se nejrůznějšími metodami pro extrakci informací z dokumentů. Aktuálně není k dispozici mnoho dostupných, kvalitních a dostatečně velkých datasetů obsahující dokumenty jako faktury, účtenky nebo podobné dokumenty. Práce zabývající se tímto problémem tedy často sbírají vlastní data a není proto jednoduché s jistotou porovnat jednotlivé techniky a modely. Drtivá většina těchto prací se zaměřuje na extrakci informací z faktur nebo účtenek a často pouze na pár nejdůležitějších polí. Tyto systémy pak spadají do jednoho ze tří níže popsaných typů jak popisuje Chiticariu et al. v práci [2].

2.2.1 Typy rozpoznávacích systému

Systémy založené na fixních pravidlech neboli rule-based systémy pro extrakci informací z dokumentů využívají množinu pravidel. Tyto pravidla jsou fixní po celou dobu běhu systému a musí být ručně vytvořena a optimalizována programátorem, který ví, jak budou vstupní dokumenty vypadat. Rule-based systémy jsou používány primárně na strukturované dokumenty s jednotnou šablonou, případně na místech, kde je jisté že se bude vyskytovat pouze několik málo předem určených šablon. Výhody rule-based systémů jsou rychlé vytvoření bez nutnosti velkého datasetu a snadné ladění. Nevýhodou může být, že systém je omezen pouze na sepsané předpisy a pro přidaní dalších šablon dokumentů musí být ručně programátorem upraveny a přidány pravidla. S větším množstvím šablon se stává optimalizování pravidel těžší a časově náročnější.

Systémy založené na strojovém učení neboli ML-based systémy namísto předem nadefinovaných pravidel, využívají vytrénované modely pro zpracování dokumentů. Modely se trénují pomocí metod strojového učení, které dokáží model připravit na různě strukturované dokumenty. Dobře navržené systémy pak pro nové šablony dokumentů vyžadují minimální rozsah manuální práce. Se změnou anotovaného datasetu a přetrénováním modelů se systém automaticky přizpůsobí na nové dosud neviděné dokumenty. Pro zpracování dokumentů je ale na rozdíl od rule-based systémů potřeba mnohem větší dataset s vytvořenými anotacemi. Dalšími nevýhodami ML-based systémů je složitější ladění a nepředvídatelné chování modelů nad neznámými daty.

Hybridní systémy jsou spojením dvou výše zmíněných systémů. V nestrukturovaných a částečně strukturovaných dokumentech může být složitější správně rozpoznat určité entity dokumentu. Entity, které mívají nepředvídatelnou délku, formu nebo pozici jsou často velmi složité na rozpoznání. Proto určité informace v dokumentu mohou být zpracovány podle nadefinovaných pravidel, zatímco zbytek dokumentu může být zpracován pomocí metod strojového učení.

[3] ML – Machine Learning – strojové učení

2.2.2 Použití jednotlivých systémů

Jak popisuje ve své práci Chiticariu et al. [2], akademická sféra již dlouhodobě upřednostňuje systémy založené na modelech strojového učení, zatímco komerční sféra je více konzervativní a převážně pak velké společnosti spoléhají více na systémy založené na fixních pravidlech. I zde se ale situace pomalu mění. A především pak společnosti zabývající se přímo extrakcí informací začínají využívat metod strojového učení.

2.2.3 Podobné vědecké práce

Práce a články zpracované na podobné téma se zabývají navržením systému pro extrakci informací z dokumentů. Pozornost v těchto pracích bývá směřována převážně na reprezentaci dokumentů, která je klíčová pro výslednou výkonnost systémů. V následující části bude popsáno několik prací, které představují různé techniky a přístupy k zpracování dokumentů. Porovnání výkonnosti systémů je sepsáno v kapitole 7.

CUTIE neboli Convolutional Universal Text Information Extractor je systém navrhnut Zhao et al. v práci [20]. CUTIE používá konvoluční neuronové sítě pro zpracování mřížky textů reprezentující dokument se zachováním vzájemných pozic. Díky přehledu o celém dokumentu dokáže CUTIE určit i závislosti dlouhého dosahu. V práci jsou navrhnuty 2 modely sítí, CUTIE-A s 67M paramatery a CUTIE-B s 14M parametry. U obou modelů byla demonstrována výkonnost na účtenkách a fakturách z různých služeb.

Majumder et al. v práci [10] navrhli systém pro extrakci informací, které jsou klasifikovány do kategorií za pomocí modelu založeného na transformátorech a self-attention mechanismu pomocí kterého je model naučen potřebné závislosti. Vstupem do modelu je vedle textových reprezentací a pozice i id entity určené pomocí Google Natural Language[4] služby. Výsledky modelu byly prezentovány na několika kategoriích informací z faktur a účtenek.

Krieger et al. v práci [7] navrhli systém založený na grafových neuronových sítích. Systém reprezentuje dokument jako graf, který je poté vstupem do modelu. Systém pracuje s textovými, sémantickými a pozičními features. Klasifikační model textové features zpracuje pomocí GRU[5], sémantické features pomocí plně propojených vrstev a výstupy spojí s pozičními features do tzv. „node features“. Node features jsou poté pro klasifikaci zpracovány pomocí Graph Attention vrstvy. Model byl poté vyhodnocen nad datasetem faktur.

CloudScan je systém navržen Palm et al. a popsán v práci [14]. Navržený systém má poskytnout nástroj pro extrakci polí z libovolných faktur bez nutnosti konfigurace nebo anotace vlastních šablon. Systém z pdf faktur vytváří n-gramy, které jsou následně zpracovány klasifikačním modelem. V práci byly navrženy dva klasifikační modely. První z nich používá pro klasifikaci logistickou regresi a druhý rekurentní neuronové sítě. Modely byly vytrénovány na 326 471 fakturách a dokáží klasifikovat informace do 8 kategorií.

[4]Google Natural Language – `https://cloud.google.com/natural-language`
[5]GRU – Gated Recurrent Unit

BERTgrid je prací [3] sepsanou Denk et al. a navrhuje systém navazující na **Chargrid** [6] sepsaný Katti et al. Oba systémy reprezentují dokumenty jako mřížku dat a zpracování probíhá pomocí konvolučních neuronových sítí.

2.2.4 Podobné komerční produkty

Jelikož je samotné rozpoznávání přepravních objednávek silně spjato s administračním systémem každé spediční či dopravní firmy, není na trhu mnoho volně dostupných nástrojů zabývající se stejným problémem. Tyto nástroje sice existují, nejčastěji však uvnitř interních systémů nadnárodních a velkých firem. Takové nástroje pak firmy neposkytují a budou optimalizované přímo na potřeby dané firmy. Ojedinělé nástroje na zpracování objednávek spedice a dopravy jsou často nabízeny bez možnosti vyzkoušení nebo ukázky. U těchto nalezených produktů, konkrétně Marine Digital[6] a PaperLessProductivity[7] nebyl nalezen ani ceník ani informace, jestli produkt poskytuje API. Další otázkou u těchto produktů je jaké informace dokáží vyhledat, jaké podporují jazyky a jaké dosahují úspěšnosti rozpoznání.

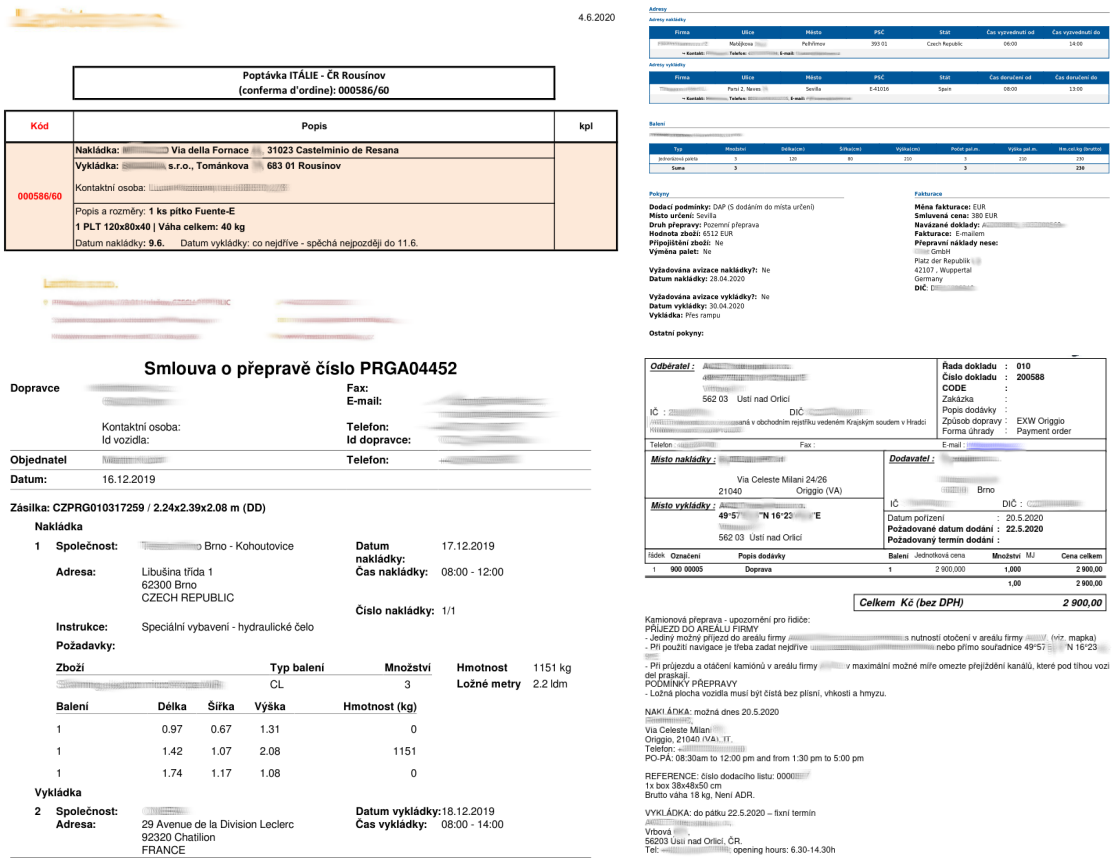
Lehce odlišný, avšak kompatibilní problém řeší produkty zaměřující se na extrakci informací z libovolných dokumentů. Jedním z nejpokročilejších v tomto poli je **Rossum**[8]. Rossum dokáže velmi spolehlivě rozpoznat pole, které se vyskytují ve většině dokumentů, jde například o datum objednávky a informace o zákazníkovi a prodejci. O poznání horší bylo rozpoznávání řádkových položek, tedy položek, které mají stejnou strukturu a nachází se na více řádcích. U přepravních dokumentů se v této formě vyskytují nejčastěji náklady, často ale nejsou vyplněny všechny hodnoty k uvedeným klíčům. U těchto dokumentů Rossum do položek zahrnul i části dokumentu následující za nákladem a celkově rozpoznání tohoto typu informací nemělo velkou úspěšnost. Testování dalších polí jako jsou adresy nebo datumy bohužel nebylo v bezplatné verzi možné. Právě pro rozpoznání dalších polí a pro odlišné šablony je nutné dokumenty ručně anotovat, popřípadě specifikovat vlastní pravidla a mechanismy přímo v aplikaci. Z testování vyplývá, že Rossum je více určené pro firmy, které používají pouze pár šablon dokumentů. V takovém případě dokáže aplikace velmi spolehlivě rozpoznávat a ulehčit zpracování dokumentů.

Obdobným komerčním produktem je **Nanonets**[9]. Z vyzkoušené bezplatné verze Nanonets dokázali spolehlivě rozpoznat čísla objednávek a ve většině případů i jména zákazníků a prodávajících. Na rozdíl od Rossum se Nanonets pokoušeli rozeznat i adresy zákazníků a prodávajících. Samotné pozice adres v dokumentu byly nalezeny přesně, ale často do ní bylo zahrnuto pouze město nebo nedostatečná část adresy. Zbytek polí Nanonets nedokázali rozpoznat vůbec, překvapivě ani identifikační čísla (IČO, DIČ) zákazníka a prodávajícího, cenu ani zboží.

Z komerčních produktů stejně jako z vědeckých článků je vidět, že hlavní soustředěnost je směřována na extrakci informací z faktur. Nástroj, který usnadní zpracování faktur má obrovsky potenciál, který cílí na všechny subjekty, které přijdou do styku s účetnictvím.

Na druhé straně přepravní dokumenty sice mají několik stejných polí s fakturami, nejdůležitější část o samotné přepravě, ale ve faktuře chybí, a právě s tou si zmíněné nástroje nedokáží moc dobře poradit. Nástroj, který by dokázal automaticky zpracovat přepravní dokumenty je pro dopravní a spediční firmy velmi ceněný, jelikož ušetří spoustu času dispečerům.

[6] Marine Digital – `https://marine-digital.com/ocr`
[7] PaperLessProductivit – `https://paperlessproductivity.com/transportation-data-capture`
[8] Rossum – `https://rossum.ai`
[9] Nanonets – `https://nanonets.com`



Obrázek 2.1: Výstřižky objednávek z datasetu.

2.3 Použitý dataset

Pro vývoj efektivního systému na extrakci informací z přepravních dokumentů je potřeba dostatek dat. Přepravní dokumenty, ale obsahují citlivé informace, které spediční a přepravní firmy nechtějí zveřejňovat, a proto neexistují žádné dostupné datasety, které by tyto informace poskytovaly.

Před počátkem vývoje jsem tedy konzultoval nejen užitečnost produktu, ale i samotný sběr dat s brněnskou spediční a dopravní firmou, která si nepřála být jmenována. Pro vývoj rozpoznávání jsem poté shromáždil dataset, který obsahuje přes 1700 unikátních objednávek. Ručně jsem anotoval 141 objednávek, ze kterých vznikl trénovací dataset o velikosti 105 objednávek a testovací dataset o velikosti 36 objednávek. Objednávky jsou datovány mezi roky 2016 až 2020 a výstřižky důležitých částí několika objednávek lze vidět na obrázku 2.1. Při procházení datasetu bylo zjištěno několik důležitých informací, které se poté výrazně promítly do vývoje samotného rozpoznávacího modulu.

94% objednávek je ve formátu pdf, zbytek ve formátu doc nebo docx. Kolem 90% dokumentů je v českém, slovenském nebo anglickém jazyce. Vyskytují se zde, ale i dokumenty v němčině, francouzštině a výjimečné i v polštině. Majorita (~70%) dokumentů obsahuje pouze jednu stranu, ostatní dokumenty obsahují nejčastěji 2-3 stránky. Nehledě na počet stran důležité informace se vyskytují na první straně, velmi vzácně některá z informací na straně druhé. Na dalších stranách se nejčastěji vyskytují podmínky přepravy, informace

o pojištění, reklamacích a dalších požadavcích, které nejsou pro tuto práci relevantní. Až na výjimky jsou objednávky částečně strukturované dokumenty, mezi výjimky patří nejčastěji dokumenty s čistě přirozeným jazykem např. snímky obrazovky z emailové konverzace. Všechny hledané hodnoty až na adresy mají určitý tvar a rozpoznatelný typ. K většině hledaných hodnot připadá jedna klíčové fráze, popř. slovo, které specifikují danou hodnotu. Specifikující fráze se v rozložení dokumentu vždy nachází nad nebo vlevo od hledané hodnoty.

Kapitola 3

Použité nástroje a znalosti

3.1 Použité nástroje

Rozpoznávací modul je napsán v programovacím jazyce Python[1] a část zdrojového kódu je v C[2] z důvodu zrychlení časově náročných operací. Klientská část webové aplikace je napsána v programovacím jazyce JavaScript[3] s využitím knihovny React[4] pro tvorbu UI komponent. Serverová část aplikace je napsána v jazyce Python s využitím frameworku Django[5].

Tensorflow je knihovna pro strojové učení, která poskytuje nejen mnoho nástrojů pro trénování modelů ale i nástroje pro jejich zkoumání a vizualizaci. Tensorflow také podporuje provádění výpočtů nejen na CPU, ale i na GPU a TPU, což může výrazně urychlit výpočetní čas.[6] V implementaci bylo využito Python API **Keras** postavené nad Tensorflow s důrazem na rychlý vývoj a experimentaci.[7]

Tesseract je knihovna pro rozpoznání textu v obrazu. Nejnovější verze Tesseract využívá LSTM neuronové sítě a podporuje celkem 116 jazyků.[8] V implementační části je používan Python wrapper **pytesseract**.[9]

Pdfminer je Python knihovna pro čtení pdf dokumentů. Knihovna vedle samotných textových elementů poskytne i informace o pozici, fontech a obrazcích nacházejících se v dokumentu.[10]

Gensim je open-source knihovna pro modelování témat a indexaci dokumentů za pomoci učení bez učitele. Knihovna podporuje širokou škálu algoritmů jako je word2vec, FastText nebo LSI a LSA.[11]

[1] Python – `https://www.python.org`
[2] C – `http://www.open-std.org/jtc1/sc22/wg14`
[3] JavaScript – `https://www.javascript.com`
[4] React – `https://reactjs.org`
[5] Django – `https://www.djangoproject.com`
[6] Tensorflow – `https://www.tensorflow.org`
[7] Keras – `https://keras.io`
[8] Tesseract OCR – `https://github.com/tesseract-ocr/tesseract`
[9] Pytesseract – `https://github.com/madmaze/pytesseract`
[10] Pdfminer – `https://github.com/pdfminer/pdfminer.six`
[11] Gensim – `https://radimrehurek.com/gensim/index.html`

Imbalanced-learn je knihovna založená na **scikit-learn**[12] a obsahuje nástroje pro vyvážení datasetu u klasifikačních problémů. Knihovna obsahuje algoritmy pro základní i pokročilé vzorkování.[13]

3.2 Reprezentace slov

Při zpracování přirozeného jazyka jsou slova základní jednotkou a informací se kterou je možné pracovat. Slova se v přirozeném jazyce skládají z posloupnosti znaků abecedy. Při strojovém zpracování slov je ale takový způsob reprezentace neefektivní, protože slova nemají fixní délku, není zachycena sémantika a je těžší provádět operace pro práci s frázemi složené z více slov. Řešením těchto problému může být převod slov na vektory. Tato technika se nazývá vektorizace. Existuje několik technik vektorizaci, kde mezi nejznámější patří:

- **One-hot encoding** pro každé nové slovo vytváří binární vektor. Vyprodukovaný vektor má velikost $1 \times N$, kde N je počet vektorizovaných slov. Pro reprezentaci věty či fráze skládající se z více slov, jednotlivé vektory zřetězíme za sebe. Tato metoda je neefektivní a při větším počtu slov se stává nepoužitelnou (u slovníku s velikostí 10 000 slov bude 99.99% elementů vektoru 0).
- **Reprezentace unikátním číslem** funguje na principu, kde každému slovu je přiřazeno unikátní číslo. Výhodou oproti předchozí metodě je že při skládání slov vzniká hustý vektor. Nevyřešenou nevýhodou ovšem zůstává, že tato metoda nezachytí žádnou podobnost mezi slovy.
- **Vnořování slov** poskytne nejen husté vektory, ale také tyto vektory vytváří tak, že podobná slova mají podobné vektory.

3.2.1 Word2vec

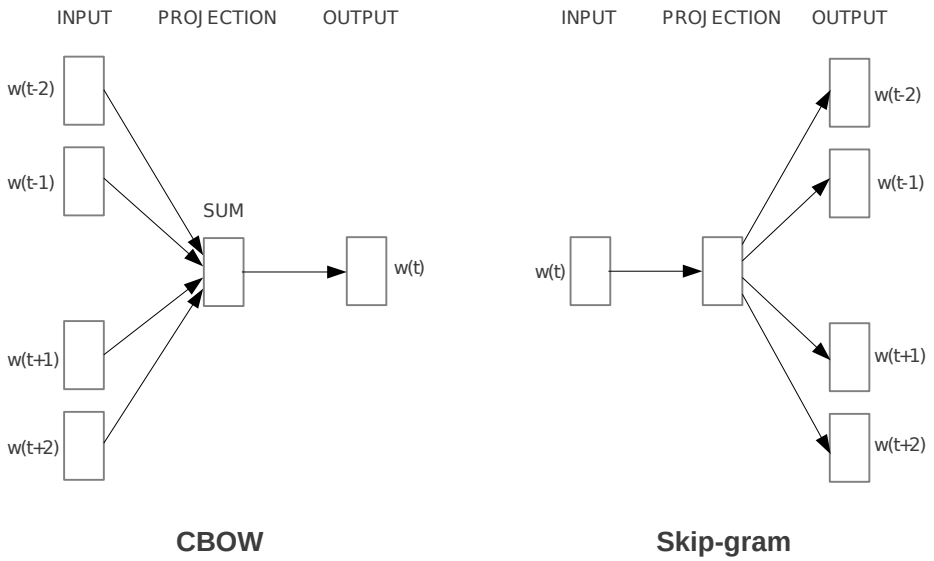
Jednou z nejznámějších a dodnes nejpoužívanějších technik vnořování slov je word2vec. Informace uvedené v této kapitole budou čerpány z původních prací, které publikoval Mikolov et al. [11, 12].

Word2vec není samostatný algoritmus, ale architektura dvou neuronových sítí a postup jejich učení. Modely se během fáze trénování učí podobnosti mezi slovy v poskytnutém textovém korpusu, ta se poté měří pomocí kosinové podobnosti.

Typy modelů

Skip-gram model předpovídá okolní slova na základě aktuální slova. Trénování probíhá na vzorcích obsahující vždy dvojici – aktuální slovo a slovo nacházející se uvnitř určitého rozsahu kolem aktuální slova. Obecně platí, že s vyšším rozsahem se zvyšuje kvalita modelu, ale také výpočetní náročnost. Slova, která jsou blíže k aktuálnímu slovu s ním častěji souvisí a jsou proto upřednostňovány častějším vybíráním do trénovacích vzorků. Skip-gram modely dokáží ve srovnání s CBOW lépe zachytit vektory vzácně se vyskytujících slov a mají lepší výkonnost na menších korpusech textu.

[12] Scikit-learn – `https://scikit-learn.org`
[13] Imbalanced-learn – `https://github.com/scikit-learn-contrib/imbalanced-learn`



Obrázek 3.1: CBOW vs Skip-gram. Převzato z práce [11]

CBOW neboli Continous Bag-of-Words model předpovídá prostřední slovo na základě jeho okolních slov. Na pořadí slov nezáleží, vektory těchto slov se průměrují nebo sčítají. Trénování je několika násobně rychlejší než u Skip-gram modelu a dokáže lépe zachytit přesnost častých slov.

Parametry modelů

Velikost vektorů určuje dimenzionalitu vektorů. Větší počet dimenzí zvyšuje do určitého bodu přesnost modelu, po určité velikosti je pro zvýšení přesnosti nutné zvýšit i počet dat. Nejčastěji se pohybuje od 20 pro menší modely až po 1000 pro trénování nad největšími korpusy textů.

Velikost klouzavého okna určuje počet zpracovávaných slov. S velikostí klouzavého okna n bude zpracováváno aktuální slovo a n slov po obou stranách.

Vzorkování nám zajistí lepší rychlost a efektivnost modelu. Při zpracovávání velkých korpusů bude trénování pomalé a může dojít k overfitting, byla proto v rámci dalšího zkoumání zpracována rozšíření, které se tyto nedostatky snaží řešit.[12]

- **Částečné vzorkování častých slov** je technika, kde každé slovo v korpusu bude s pravděpodobností $P(w_i)$ během trénování zahozeno. $P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$, kde t je zvolený práh obvykle kolem 10^{-5} a $f(w_i)$ je frekvence slova w_i ve zpracovávaném korpusu.
- **Negativní vzorkování** pro každý trénovací vzorek je vzato následující slovo jako pozitivní vzorek a k náhodných slov ze slovníku jako negativní vzorky. Trénování pak namísto předpovídání pravděpodobností na výskyt v kontextu pro všechny slova ve slovníku, probíhá jako binární klasifikace pro vytvořené vzorky. Nejefektivnější počet negativních vzorků k je 5-20 pro menší datasety a 2-5 vzorků pro větší datasety.

Implementací částečného a negativního vzorkování se nejen zrychlí trénování modelů, ale také zvýší přesnost naučených vektorů, zvláště pak slov, které se v korpusu vyskytují jen vzácně.

Metriky podobnosti slov

Euklidovská vzdálenost je metrika pro měření vzdálenosti dvou bodů v euklidovském prostoru. V rovnici 3.1 je definována euklidovská vzdálenost mezi vektory A a B.

$$d(A, B) = \sqrt{\sum_{i=1} n(A_i - B_i)^2}. \tag{3.1}$$

Kosinová podobnost anglicky cosine similarity je metrika podobnosti mezi dvěma nenulovými vektory, která měří kosinus úhlu mezi dvěma vektory. Je nejčastěji používanou metrikou pro změření podobnosti mezi slovy. Rovnice 3.2 definuje kosinovou podobnost mezi vektory A a B.

$$sim(A, B) = cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}. \tag{3.2}$$

Word Mover's Distance zkráceně WMD je technika využívaná u vnořování slov, samotný výpočet je často prováděn nad dokumenty slov. Vzdálenost těchto dokumentů se spočítá jako nejkratší vzdálenost, která je potřeba aby se vektory jednoho dokumentu dostali na pozice vektorů druhého dokumentu.

3.2.2 Analýza hlavních komponent

Analýza Hlavních Komponent anglicky Principal Component Analysis zkráceně PCA je transformační metoda, která se často používá pro redukci dimenzionality dat s co nejmenší ztrátou informací. Pro takovou transformaci jsou využity kovarianční matice, kde pro d dimenzionální data je vytvořena $d \times d$ kovarianční matice, kde každý prvek popisuje kovarianci mezi dvěma proměnnými, matice je tedy symetrická. Převážně pak ve financí se místo kovarianční využívá korelační matice, která je ale v případě standardizovaných dat stejná jako kovarianční. Pomocí matice jsou pak vypočteny vlastní vektory (eigenvectors) a vlastní čísla (eigenvalues). Vlastní vektory mají délku 1 a udávají směr nové osy, zatímco vlastní čísla udávají velikost. Jádrem PCA je pak samotná redukce, která probíhá odstraněním některých hlavních komponent (principal components). Vlastní čísla udávají velikost, a tedy i váhu jednotlivých vlastních vektorů, výběr tedy probíhá seřazením vlastních čísel a následně je vybráno k největších (nejvlivnějších) vlastních čísel. Ty spolu s jejich vlastními vektory tvoří nový redukovaný prostor.

Kroky algoritmu:

- Standardizace dat
- Vypočtení kovarianční nebo korelační matice pro data
- Vypočtení vlastních vektorů a čísel
- Výběr hlavních komponent

3.3 Klasifikační metody a modely

Klasifikace je proces ve strojovém učení a statistice během kterého jsou vstupní data pomocí heuristik a aproximace funkcí rozdělovány do několika tříd neboli kategorií. Ve strojovém učení je trénování klasifikačních modelů prováděno s učitelem, během trénování se model pomocí anotovaných dat snaží najít optimální funkci na rozdělení vstupních dat do kategorií. Existuje několik typů klasifikace, mezi hlavní patří klasifikace binární, která se zabývá kategorizací dat do dvou tříd. Unární klasifikace se zabývá identifikací dat spadajících pod jednu kategorii mezi daty různých kategorií. Vícetřídní neboli *multiclass classification* klasifikuje vstup do jedné ze tří a více kategorií. Víceštítková *multilabel classification* klasifikace kategorizuje vstup do všech vhodných kategorií (nejčastěji reprezentováno vektorem). Některé metody a modely dokáží kategorizovat s určitou pravděpodobností, takovou klasifikaci pak nazýváme pravděpodobnostní.

Obdobným procesem je regrese zabývající se odhadem hodnoty určité náhodné veličiny a shlukování, které data třídí do skupin dat, které jsou si nejvíce podobné, bez trénovacích dat.

Logistická regrese

Logistická regrese je statistická metoda modelující vztah mezi predikátory a výstupní kategorickou proměnnou (třída), podle které je logistická regrese dělená na binární (dvě třídy) a multinomické (více tříd). Základní a nejčastěji používaný je binární model:

$$P(A) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 X_1 + ... + \beta_n X_n))}, \tag{3.3}$$

kde:

$P(A)$ je pravděpodobnost, že vstupní data patří do určité kategorie.

n je počet vstupních proměnných.

β jsou parametry modelu.

X jsou vstupní proměnné.

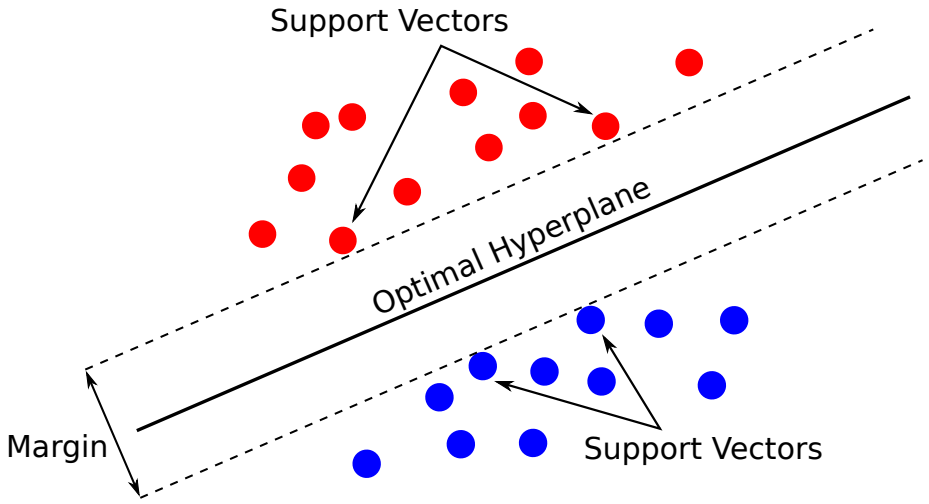
Parametry modelu se nastavují pomocí dostupných dat nejčastěji metodou maximální věrohodnosti. Výhodou použití logistické regrese ve strojovém učení je jednoduchá implementace a vysoká rychlost nejen učení, ale i predikcí.

Naive Bayes

Naive Bayes je skupina metod používaná pro binární i vícenásobnou klasifikaci. Výpočet pravděpodobností pro kategorizaci je založena na Bayesově teorému a zároveň klasifikátory předpokládají nezávislost mezi vstupními parametry. Jedná se o zjednodušení, které v reálném světě většinou neplatí ale je nutné, aby bylo možné sestrojit modely s velkým počtem vstupních proměnných. Naive Bayes metody se i přes určité zjednodušení předpokladů ukázaly jako velice efektivní a jsou využívány na filtraci spamů nebo třídění dokumentů. Naive Bayes pro vstupní parametry $x_1, ..., x_n$ a výstupní třídu y definujeme jako

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3.4}$$

a pomocí MAP (maximum a posteriori) je poté odhadnuto $P(y)$ a $P(x_i|y)$. Existuje několik typů Naive Bayes modelů, liší se především distribuční funkcí $P(x_i|y)$.



Obrázek 3.2: Support Vector Machines. Na obrázku jsou vzorky dvou kategorií oddělené optimální nadrovinou. Vzorky jednotlivých kategorií nacházející se nejblíže optimální nadrovině jsou nazývány podpůrné vektory.

Support Vector Machines

zkráceně SVM je množina metod strojového učení s učitelem používané pro klasifikaci a regresi. SVM model prokládá nadrovinu v N dimenzionálním prostoru tak aby byly data co nejlépe odděleny, tato rozhodovací nadrovina poté slouží pro klasifikaci. Pro nalezení ideální nadroviny mají hlavní vliv podpůrné vektory. Podpůrné vektory jsou datové body ležící nejblíže rozhodovací nadrovině a které jsou nejhůře klasifikovatelné. SVM poté hledá nadrovinu, která má největší okraj od podpůrných vektorů, taková nadrovina se nazývá optimální. SVM lze využít i pro nelineární klasifikaci za pomoci kernel triku, který zobrazuje vstupní prostor hodnot na prostor ve větší dimenzi bez nutnosti přepočtu souřadnic jednotlivých bodů.

Výhodou SVM je efektivnost ve velkém počtu dimenzí i při malém počtu dat, základní verze SVM ovšem neposkytuje pravděpodobnostní klasifikaci.[17]

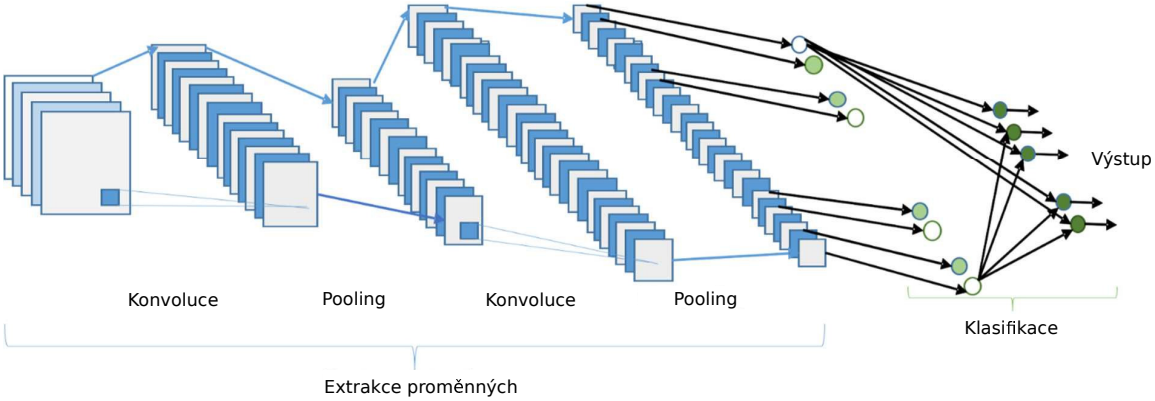
Konvoluční neuronové sítě

Konvoluční neuronové síťě (CNN) jsou podmnožinou hlubokých neuronových sítí, většinou využívané pro počítačové vidění, zpracování zvuku nebo NLP. CNN se liší od klasických umělých neuronových sítí tím, že jejich architektury mají vlastnosti, které dokáží vyextrahovat užitečné informace ze vstupních dat. Při vytváření architektury CNN se používají tři hlavní typy vrstev. Konvoluční, polling a plně propojené vrstvy.

Konvoluce a křížová korelace

Základem CNN jak už z názvu vyplývá je konvoluce. Konvoluce je matematická operace zpracovávající dvě funkce a definovaná vztahem

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(t)w(t-a)da. \tag{3.5}$$



Obrázek 3.3: Architektura konvoluční sítě. Převzato z práce [1]

Během strojového učení funkce x odpovídá **vstupním datům** např. obraz nebo sekvence slov, funkce w se pak v CNN terminologii nazývá **kernel** a výstup **feature map**. Konvoluce je v knihovnách pro strojové učení často implementována jako křížová korelace, která je podobná konvoluci, ale nepřevrací kernel. Obě funkce se pak často nachází v diskrétní a více dimenzionální podobě. Diskrétní křížovou korelaci pro 2D vstupní data I s 2D kernelem K, pak definujeme jako

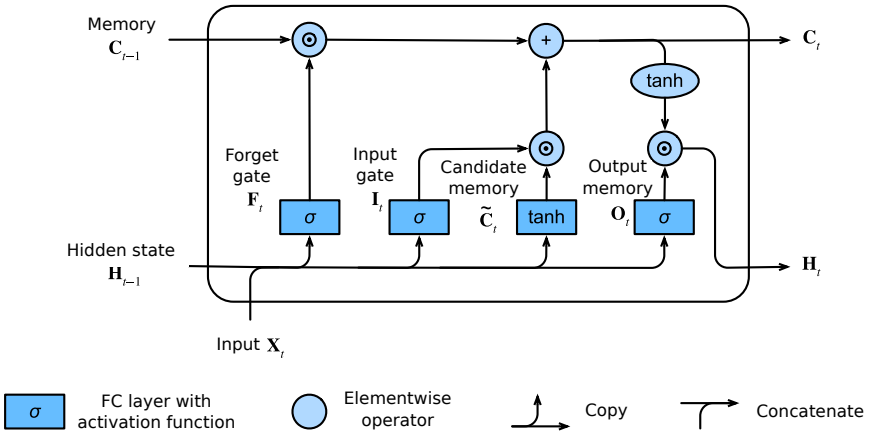
$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n). \tag{3.6}$$

Konvoluční vrstvy

U klasických neuronových sítí jsou používané plně propojené vrstvy. S narůstající komplexitou sítě jako je zpracování obrazu nebo zvuku, ale klesá efektivita těchto modelů a je nutné udržovat zbytečné množství vah. Konvoluční vrstvy propojují neurony pouze na lokální úrovni určené velikostí kernelu, rozsah této úrovně se nazývá **receptive field**.

Výstupní počet neuronů je poté závislý na 3 hyperparametrech. **Hloubka** neboli počet filtrů, kde každý může být pro jinou funkcionalitu např. detekce hran nebo barev. Dalším hyperparametrem je **stride**, který určuje posun filtru v matici dat, čím větší je stride tím menší je výstup. Další možností, jak ovlivnit velikost výstupu je vyplnit okolí vstupu nulami, tato metoda se nazývá **zero padding** a souvisejícím hyperparametrem můžeme ovlivnit velikost výplně.

Další výhodou konvolučních vrstev je sdílení parametrů, to vychází z předpokladu, že určité naučené váhy jsou užitečné napříč celým vstupem. V praxi to pak vypadá tak, že neuronová síť se neučí rozdílné váhy filtru pro každou pozici vstupu, ale je vytrénována jedna množina vah, která může detekovat například hrany. Tento filtr je pak používán pro každou pozici vstupu. Díky sdílení parametrů je možné dosáhnout ekvivariance vrstvy, tedy pokud se změní vstup, je stejným způsobem změněn i výstup. Pokud je tedy v obraze posunut hledaný objekt bude stejným způsobem posunuta i jeho reprezentace ve výstupu vrstvy. V konvoluční vrstvách ovšem není zachována ekvivariance pro všechny operace jako je rotace nebo změna měřítka obrazu.



Obrázek 3.4: LSTM architektura. Převzato z práce [19]

Pooling vrstvy

Pooling vrstvy se běžně vyskytují za konvoluční vrstvou, kde dále upravují její výstup za pomocí pooling funkce. Pooling funkce provádějí statistické operace nad rámcem dat o určité velikosti. Mezi klasické operace patří průměrování nebo vybírání maxima. Výstupem pooling vrstev jsou pak výsledky těchto funkcí. CNN se za pomocí pooling vrstvy stávají přibližně invariantní, tedy pokud jsou vstupní data mírně posunuta, výstup zůstává stejný.[8, 4]

Rekurentní neuronové síťe

Rekurentní neuronové sítě zkráceně RNN jsou neuronové sítě využívané ke zpracování sekvenčních dat. Podobně jako konvoluční i rekurentní neuronové sítě si dokáží poradit se daty, které by pro klasické neuronové sítě byly příliš velké. Na rozdíl od klasických sítí mohou RNN také využít svůj interní stav pro efektivní zpracování různě dlouhých sekvencí, kde na sebe mohou být jednotlivé vzorky vzájemně závislé. Tato vlastnost je ceněná obzvláště u zpracování signálu a NLP problémů. RNN se ovšem oproti klasickým neuronovým sítím těžce trénují a často u nich dochází k mizivému (vanishing) a explodujícímu (exploding) gradientu. První jev nastává, když parciální derivace chybové funkce je příliš malá a váhy sítě se tak upraví pouze mizivě. Ve druhém případě se chyba gradientu akumuluje a dosáhne hodnot při kterých není síť stabilní a nedokáže se učit z trénovacích dat.[15]

Long Short Term Memory neboli LSTM je speciální architektura RNN, která se dokáže učit dlouhodobé závislosti a řeší problémy s vanishing a exploding gradienty. LSTM se skládá z několika vnitřních mechanismů nazývaných brány (gates) a stavu buňky. Mezi tyto brány patří **forget gate** která rozhoduje, jaké informace jsou důležité a budou ponechány a které budou zahozeny. **Input gate** se skládá ze dvou vrstev a rozhoduje o kolik se bude upravovat stav buňky. Samotný **stav buňky** se vypočítá vynásobením předchozího stavu buňky a výstupem z forget gate a výsledek je dále sečten s výstupem z input gate. **Output gate** pak společně s novým stavem buňky rozhoduje o výstupu ze sítě.

Porovnání CNN a RNN pro NLP

Ve studii [18] Yin et al. systematicky porovnali výkonost CNN a RNN (konkrétně LSTM a GRU) na reprezentativním vzorku NLP úkolů. Výsledky podle očekávání ukázali, že RNN sítě si dokáží mnohem lépe poradit s dlouhými texty a tam kde jsou zapotřebí sémantické závislosti dlouhého dosahu. Naopak u problémů vyžadující rozpoznání lokálních klíčových frází jsou efektivnější CNN. CNN jsou také nepatrně lepší u úkolů s texty pod 10 slov, nad touto hranicí získávají výhodu RNN.

3.3.1 Vícetřídní klasifikace s využitím binárních klasifikátorů

Obecně jsou vícetřídní klasifikátory těžší na vytrénování, obzvláště pak s malým počtem trénovacích vzorků. Navíc některé klasifikátory neumí řešit vícetřídní klasifikace, a proto existují techniky jak rozdělit vícetřídní problém na několik binárních problémů.

One-vs-All nebo také One-vs-Rest je metoda, která v případě N kategorií, rozdělí jeden vícetřídní klasifikační problém do N binární problémů/datasetů. V každém datasetu jsou vzorky jedné kategorie označeny jako pozitivní a vzorky všech ostatních kategorií jsou brány jako negativní.

One-vs-One je metoda, kde je vytvořeno $N \cdot (N-1)/2$ binárních datasetů. Datasety jsou vytvořeny pro všechny možné páry kategorií a každý dataset pak obsahuje vzorky pouze jednoho páru kategorií.

3.3.2 Nevyvážené datasety

Častým problémem u klasifikačních problémů jsou nevyvážené datasety, které vznikají nerovnoměrným počtem vzorků jednotlivých tříd. Klasickým příkladem může být rozpoznávání spamů mezi emailové komunikaci. Pokud je spam pouze jeden mail z tisíce, model se naučí klasifikovat všechny maily jako neškodné. Úspěšnost takového modelu je pak velmi vysoká ovšem samotný model je nepoužitelný. Existuje tedy několik technik jak takový problém řešit.[13]

Podvzorkování je metoda při které jsou odstraněny některé prvky majoritní třídy pro vyrovnání počtu vzorků.

- **Náhodné podvzorkování** vzorků majoritní třídy je odstranění náhodných prvků, což obzvláště u malých datasetů může vést k ztrátě důležitých informací.
- **Near Miss** je skupina metod snažící se minimalizovat risk ztráty důležitých vzorků. Funguje na principu vzdálenosti mezi vzorky majoritní a minoritní třídy.

Převzorkování je metoda, která pro vyvážení datasetu vytváří nové vzorky do minoritní třídy.

- **Převzorkování náhodných prvků** duplikuje prvky v minoritní kategorii. Pouze vyrovná počet vzorků, ale nepřináší žádné nové informace.

- **SMOTE** neboli Synthetic Minority Oversampling Technique je algorimus, který využívá KNN[14]. SMOTE náhodně vybere vzorek minoritní třídy a pomocí KNN najde jeho k nejbližších sousedů stejné třídy. Náhodný prvek je poté spojen s nalezenými sousedy a vektory spojující tyto body jsou vynásobeny náhodnými čísly v intervalu (0, 1). Nově nalezené pozice jsou nazývány syntetické body a jsou přidány do datasetu.
- **ADASYN** neboli Adaptive Synthetic Sampling Approach je algoritmus založený na SMOTE, který klade důraz na vytváření nových vzorků na základě obtížnosti jejich klasifikace.

Váhování tříd je další možností jak vylepšit výkonost modelu trénovaným nad nevyváženým dataset. Princip spočívá v udělení větší penalizace za špatnou klasifikaci minoritní třídy nebo naopak udělení větší odměny za správnou klasifikaci minoritní třídy. Tímto způsobem je model donucen klást větší důraz na správnou klasifikaci minoritních vzorků.

3.3.3 Hodnocení klasifikačních modelů

Při binární klasifikaci jsou 4 možné výstupy:

- **True positive** zkráceně TP jsou pozorování při kterém je modelu poskytnut vstup, který patří a zároveň je i modelem zařazen do určité kategorie.
- **True negative** zkráceně TN jsou pozorování při kterém je modelu poskytnut vstup, který nepatří a ani není modelem zařazen do určité kategorie.
- **False positive** zkráceně FP jsou pozorování při kterém je modelu poskytnut vstup, který nepatří do určité kategorie, ale je do ní zařazen modelem.
- **False negative** zkráceně FN jsou pozorování při kterém je modelu poskytnut vstup, který patří do určité kategorie, ale není do ní zařazen modelem.

3.3.4 Výkonnostní metriky klasifikačních modelů

Matice záměn anglicky confusion matrix je matice do které jsou ukládány predikce modelu. Z matice se dále počítají další metriky výkonosti modelu.

	Negativní predikce	Pozitivní predikce
Skutečně negativní	TN	FP
Skutečně pozitivní	FN	TP

Tabulka 3.1: Matice záměn

Přesnost anglicky accuracy udává jaké procento vzorků model správně predikoval.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

[14]KNN – Algoritmus k-nejbližších sousedů

Preciznost anglicky precision někdy také jako PPV (Positive Predictive Value) je podíl počtu správně zařazených vzorků do kategorie oproti všem vzorkům.

$$precision = \frac{TP}{TP + FP} \tag{3.8}$$

Senzitivita anglicky recall nebo také TP (True Positive Rate) je podíl počtu správně zařazených vzorků do kategorie oproti všem vzorkům, které do kategorie opravdu patři.

$$recall = \frac{TP}{TP + FN} \tag{3.9}$$

Specificita udává kolik prvků nezařazených do kategorie do ní opravdu nepatři.

$$specificity = \frac{TN}{TN + FP} \tag{3.10}$$

Fall-out neboli FPR (False Positive Rate), udává pravděpodobnost nesprávného určení u vzorků, které nepatří do kategorie.

$$fallout = \frac{FP}{FP + TN} \tag{3.11}$$

F1 skóre je harmonický průměr přesnosti a senzitivity.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3.12}$$

AUC ROC je metrika, která dokáže zachytit výkonost modelu s různými klasifikačními prahy. AUC (Area Under the Curve) znamená plochu pod křivkou a je nejčastěji používána pravě s křivkou ROC (Receiver Operating Characteristic), která se skládá z metriky FPR na ose x a TPR na ose y. Pro vygenerování funkce pod kterou se bude plocha počítat je vygenerováno x bodů v ROC prostoru, pokaždé s jiným klasifikačním prahem.

Kapitola 4

Návrh řešení

4.1 Rozpoznávání objednávek

Modul pro rozpoznávání objednávek je hybridní systém pro extrakci informací z přepravních dokumentů. Celý proces zpracování objednávky lze najít na obrázku 4.1. V následující části bude popsán jeho návrh, architektura i postupy. Díky rozmanitosti zpracovávaných objednávek i samotných kategorií nebylo možné použít přesné postupy ze článků zabývající se podobnými problémy. Bylo proto nutné provést určité množství experimentů pro nalezení „ideálního“ návrhu. V této části budou tedy popsány i relevantní experimenty, které by měly poukázat na slepé uličky, kterými by se vývoj dále neměl ubírat a stejně tak budou popsány postupy, které by mohly být užitečné a efektivní v pozdější fázi produktu.

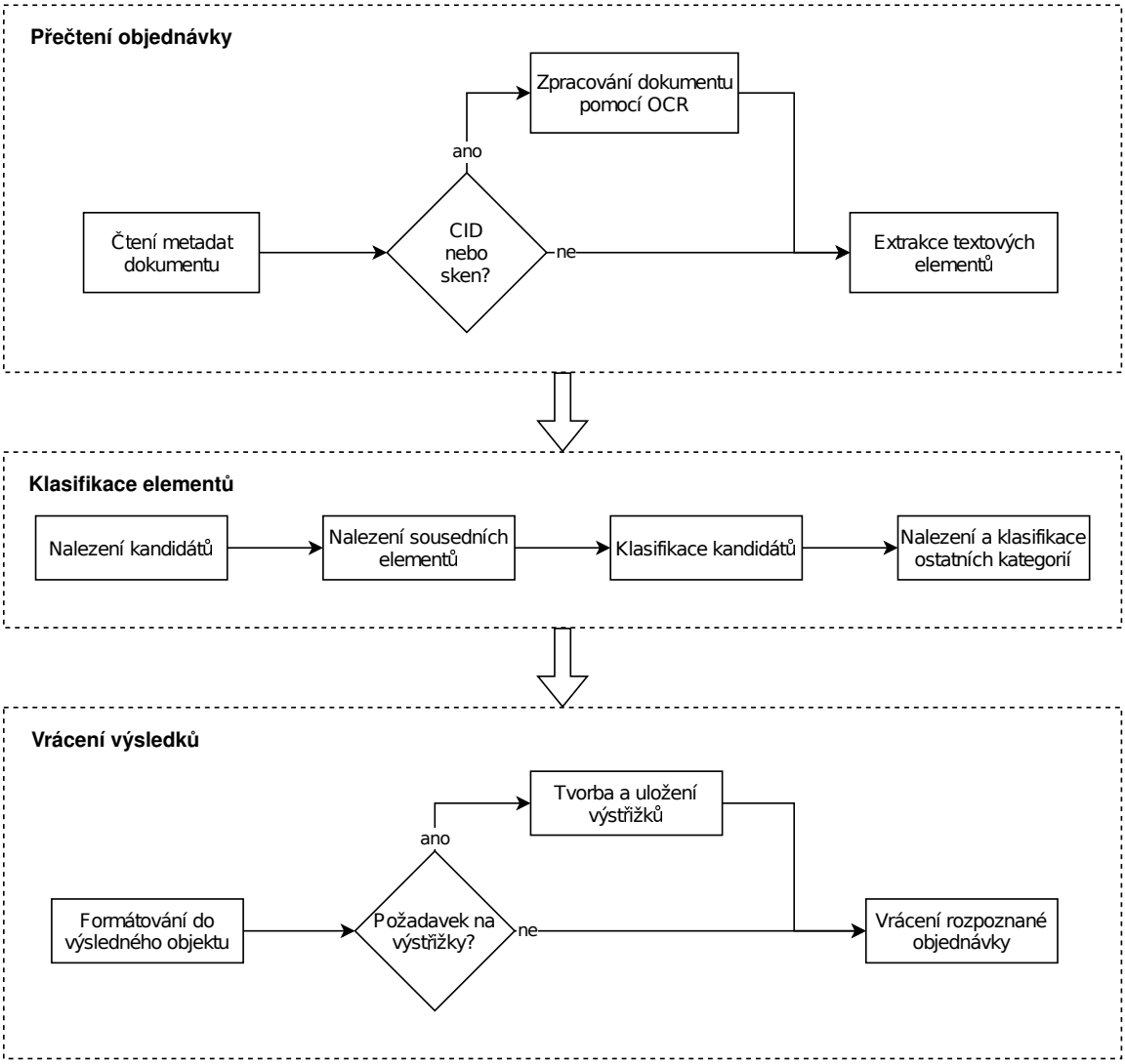
4.1.1 Čtení a předzpracování objednávky

Vstupem pro každé rozpoznávání je objednávka na přepravu ve formátu pdf. Pokud byl dokument digitálně zpracován a obsahuje textové elementy, jsou tyto elementy postupně čteny a společně s jejich pozicí ukládány. V případě, že dokument neobsahuje textové elementy nebo obsahuje CID[1] fonty, které jsou proprietární nebo jsou neznámé na běžícím systému, musí se dokument zpracovat pomocí OCR. Pytesseract poskytuje zpracování pouze pro obrázky, vstupní dokument je tedy převeden na obrázek. Oskenované dokumenty jsou často ve vysokém rozlišením, jejich převod v plné kvalitě by tak byl výpočetně i časově náročný, proto převod probíhá s fixním DPI[2]. Experimentálně bylo zjištěno, že ideální hodnota DPI je 125, taková hodnota zajistí rychlý převod, ale také dostatečnou kvalitu pro další čtení a zpracování. Převedený obrázek je pomocí OCR převeden zpět na pdf, nyní již s dostupnými texty. Z takového pdf jsou přečteny textové elementy spolu s jejich pozicemi.

4.1.2 Zpracování a rozpoznání objednávky

Další částí je samotné rozpoznávání, na jehož začátku proběhne nalezení potenciálních kandidátů. Kandidát je textový element obsahující hodnotu, kterou lze přiřadit k jedné z kategorií ze schémy objednávky. Schéma objednávky je JSON objekt, který definuje všechny kategorie k identifikaci a jejich dodatečné informace jako barva výsledného zvýraznění. Každá kategorie má své unikátní jméno a entitu. Mezi základní podporované entity patří

[1] CID – Character Identifier font – Každý znak je unikátně zakódován, kódování se liší mezi typy fontů
[2] DPI – Dots per inch – hodnota udávající počet pixelů v jednom palci



Obrázek 4.1: Pipeline aplikace. Proces zpracování objednávek se dělí do 3 částí. V prvním je objednávka přečtena a vrací nalezené textové elementy. Ve druhé části se textové elementy zpracují a naleznou se klíčové hodnoty. V poslední části se z dostupných informací vytvoří finální JSON objekt a v případě požadavku se vytvoří výstřižky.

datum, adresa, cena, dále pak typy k identifikaci společnosti (jméno, ičo a dič) a k informacím o převáženém nákladu (rozměry, kvantita a váha).

Všechny kategorie kandidátů jsou až na adresy hledány podle regulárních výrazů a případných specifických filtrujících heuristik. U všech potenciálních kandidátů se poté naleznou sousední textové pole. Jak bylo popsáno v sekci 2.3, klíčová slova nebo fráze které popisují hodnotu se nachází nad nebo nalevo od hodnoty, stačí tedy nalézt pouze tyto sousední prvky. Pokud je sousední prvek také potenciální kandidát, může se jednat o tabulková data, a proto na každou stranu hledáme první sousední prvek, který není kandidát, dojde tak k zvýšení pravděpodobnosti nalezení nějaké specifikujícího textu.

Nejen sousední prvky, ale i samotný kandidátní prvek může obsahovat důležité informace pro následující klasifikaci, nejčastěji se tak děje u objednávek obsahující více přirozeného

textu. Je proto nutné z textu kandidáta odstranit samotnou kandidátní hodnotou nalezenou regulárními výrazy. Takto zpracovaný kandidátní text je spolu se sousedními prvky převeden na normalizované vektory pomocí word2vec modelů a s pozicí kandidátního prvku je poté ohodnocen klasifikačními modely. Pro hledanou kategorii je takto vybrán nejlépe klasifikovaný kandidát. Z rozpoznaných informací je vytvořen finální JSON objekt reprezentující strojově zpracovanou objednávku.

Výše popsaný mechanismus byl výsledkem několika experimentů, během kterých byl zkoumán především vliv výběru soudních prvků, který se ukázal jako klíčový pro efektivní klasifikaci. Jako neefektivní se ukázala metoda při které byly za sousední prvky považovány všechny textové elementy v určité oblasti od kandidáta. Obecně platilo, že čím větší oblast byla vybrána, tím obtížnější byla následná klasifikace. Stejně neefektivní se ukázal i návrh, při kterém byly vybírány sousední prvky ze všech stran. Přidaná hodnota těchto prvků byla v drtivé většině případů nulová a došlo tak pouze k zhoršení klasifikace.

4.1.3 Hledání kandidátů

Jak bylo výše zmíněno kandidáti, tedy textové elementy v dokumentu, které mají potenciál stát se hodnotou k některé z kategorií v objednávce se hledají převážně podle regulárních výrazů. Ideální rozpoznávací systém by měl spoléhat na takové klasifikační modely a metody tak, že pokud rozšíříme dataset, systém se automaticky přizpůsobí. Regulární výrazy jsou ale fixní pravidla, která se spolu s datasetem nijak nevyvíjí a do určité míry tak mohou omezovat výkonnost systému. Pro regulární výrazy existují podle kontextu různé alternativy, které ale nemusí být nutně lepší volbou.

Detekce ceny

Mezi kategorie, kde regulární výrazy dokáží spolehlivě zachytit hodnoty dané kategorie je cena. Jediným vodítkem k jistému určení je měna, která se ale ne vždy vyskytuje v bezprostřední blízkosti hodnoty a je proto efektivnější označit za kandidáty na cenu všechny číselné hodnoty a hledanou cenu určit až po predikci klasifikátorů.

Detekce IČO a DIČ

Podobně lehce detekovatelné kategorie s pomocí regulárních výrazů jsou IČO a DIČ, mezinárodně nazývané jako *VAT Identification Number*[3]. Pro tyto kategorie existují formáty specifikující tvar, kterých mohou hodnoty nabývat. Tyto hodnoty se pak často validují dodatečným výpočtem, ten ale můžeme vynechat, protože nevalidní hodnoty splňující formu se v dokumentech až na výjimky neobjevují. Výjimky pak často patří do jiné kategorie, tato skutečnost je pak ale promítnuta v hodnocení klasifikátorů.

Detekce názvů společností

O poznání horší je detekce názvů subjektů. Názvy nemají žádnou specifickou formu ani délku, která by se dala využít. Lze ale využít skutečnosti, že přepravy nejsou provozovány ani poptávány fyzickými osobami. Jak dodavatel, tak zákazník jsou tedy společnosti s určitým typem společnosti. V Česku jsou nejznámější typy společností s.r.o. nebo a.s. podobné pojmenování pak existují i v ostatních státech. V objednávkách pak firmy uvádějí nejen

[3]VAT Identification Number – Value Added Tax Identification Number

své názvy, ale právě i typ podle kterého lze sestavit regulární výraz detekující právě názvy společností.

Jednou z alternativou pro hledání názvů společností je Google Natural Language API[4]. API využívající modely vytrénované na obrovských korpusech dat má ovšem hned několik problémů. Objednávky rozpoznávané v rámci této práce neobsahují pouze přirozený text a často se v nich uvádí informace, které nejsou veřejně probírané a nevyskytují se ve zmíněných korpusech. To způsobí, že API nedokáže detekovat názvy menších ani středních společností. V neposlední řadě API zatím nepodporuje český jazyk a v případě frekventovaného používání je nutné platit za jednotlivé dotazy.

Detekce datumů

Samotné datum může na první pohled vypadat jako lehce detekovatelné i za pomocí regulárních výrazů. V objednávkách se bohužel vyskytují nejrůznější formy datumů, oddělené různými znaky, často zkrácené a občas se místo specifického datumu vyskytují také intervaly nebo slovní spojení. Takové výrazy jsou pro regulární výrazy složité na detekci, a proto jsou do systému zavedeny určité kompromisy.

Pro detekci datumů existují různé nástroje, jedním z nejrozšířenějších je dateparser[5]. Dateparser obsahuje slovníky pro práci s daty, takže je schopný detekovat i slovní fráze. Velkou nevýhodou je ovšem špatná detekce datumů s evropským formátem. Například „1.8.“ není nástrojem vyhodnoceno jako datum. Tato nevýhoda se dá odstranit dodatečnými regulárními výrazy, což ale snižuje rychlost a zvyšuje komplexitu systému, zatímco přidaná hodnota je spíše minimální.

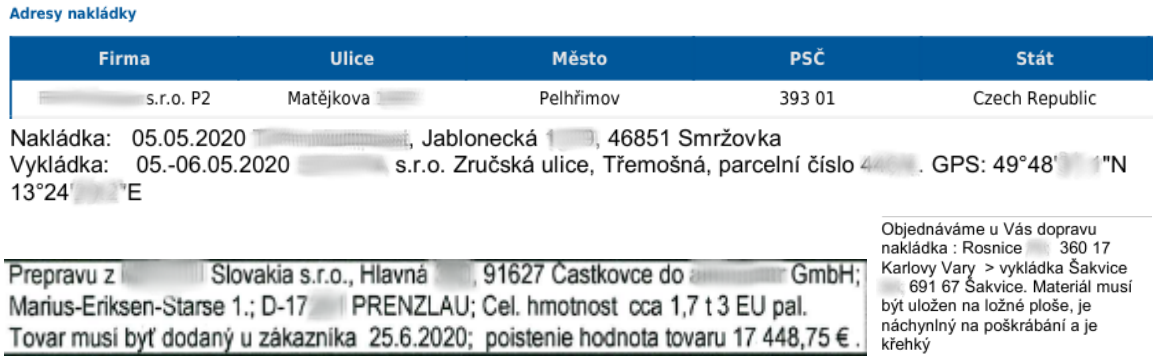
Detekce adres

Adresy jsou odlišné od jiných entit především svou nepředvídatelnou formou, délkou i pozicí. Adresy jsou někdy roztaženy na několik řádků, jindy jsou naopak všechny důležité adresy sepsány v jedné větě. Některé jejich příklady lze vidět na obrázku 4.2. Je velmi složité a neefektivní rozpoznávat adresy pravidly nebo pomocí vzorů. Určitou úspěšnost má regulární výraz na PSČ, bohužel adresy často PSČ neobsahují, navíc by regulární výraz musel pokrýt i nejrůznější zahraniční formy. Značně spolehlivějším řešením se ukázalo najít v dokumentu všechny města a poté za adresu považovat jejich okolí.

Projekt OpenStreetMap[6] obsahuje velké množství volně dostupných geodat, mezi které patří nejen podklady pro mapy, obrázky z ulic, ale i databáze míst. Na všechny města, vesnice i malé usedlosti určité oblasti se pak lze dotazovat přes OpenStreetMap API[7]. Odpověď z API obsahuje vedle základních informací o místech i všechny jejich známé formy názvů i s překlady. Takto vytvořená offline databáze obydlených míst v celé Evropě, čítá necelých 600 000 názvů míst.

Za pomoci této databáze je nyní možné nalézt textové prvky, které obsahují názvy měst. Mezi názvy měst můžeme ale narazit i na výrazy běžně používané v přirozeném jazyce, a proto jsou jednoduchou heuristikou textové elementy filtrovány. Pro zpracování přepravních objednávek jsou pak důležité zejména dvě adresy, a to adresa nakládky a vykládky, které často doplňují a nacházejí se v blízkosti datumů nakládky a vykládky. Jako spolehlivou metodou se ukázalo klasifikovat adresy podle jejich vzdálenosti k datům.

[4]Google Natural Language API – `https://cloud.google.com/natural-language`
[5]dateparser – `https://dateparser.readthedocs.io/en/latest`
[6]OpenStreetMap – `https://www.openstreetmap.org`
[7]OpenStreetMap API – `https://wiki.openstreetmap.org/wiki/API`



Obrázek 4.2: Ukázky adresy z objednávek. Jak lze vidět některé adresy jsou strukturované do tabulek jiné do přirozené textu. Nezřídka kdy se také stává, že se jak adresa nakládky tak i vykládky vyskytují v jedné větě nebo odstavci.

Detekce ostatních entit

V objednávkách se mimo výše zmíněné entity vyskytují i informace o nákladu do kterých patří rozměry, kvantita a váha. Rozměry mohou být uvedeny buď jako jednotná informace, tedy čísla oddělené nejčastěji znakem „x“ nebo se nacházejí odděleně například v tabulce, kde hodnoty náleží číselným hodnotám v buňkách. V obou případech je snadné tyto informace detekovat pomocí regulárních výrazů. Podobným způsobem jde detekovat i kvantita a váha nákladu. Doplňující informace o nákladu jako je jednotka rozměru nebo váhy lze většinou najít poblíž nalezených číselných hodnot. Do budoucna lze detekci rozšířit o další informace o nákladu jako je možnost stohovatelnosti nákladu nebo ADR[8]. Tyto informace mají binární hodnotu a pro jejich nalezení budou stačit regulární výrazy s jednoduchou heuristikou.

4.1.4 Vytváření kontextových výstřižků

Posledním krokem zpracování objednávky je vytvoření kontextových oken neboli výstřižků. Tyto výstřižky jsou tvořeny ze vstupního dokumentu a vždy zachycují určitou skupinu informací. Existuje 5 takových jsou obsaženy všechny rozpoznané hodnoty dané kategorie. U každého kontextu jsou také zvýrazněny rozpoznané hodnoty, což pomůže uživateli snáze zkontrolovat rozpoznané údaje.

4.2 Word2vec modely

Word2vec modely mají zásadní dopad na efektivnost učení a výslednou úspěšnost klasifikačních modelů, v rámci vývoje byl tedy kladen velký důraz na nalezení optimální architektury a procesu trénování modelu. Během experimentace byly vyzkoušeny obě techniky word2vec, u každé techniky byl zkoumán vliv hyperparametrů na výsledný model. Podobné experimentace s word2vec již proběhly a byly sepsány v práci [12]. Tyto pokusy ovšem vždy probíhaly na korpusu přirozeného textu a jak se později ukázalo, určité jevy neplatily při korpusu textů z částečné strukturovaných dokumentů. Proto vedle samotných modelů byl také zjišťován vliv a možné zlepšení výkonnosti modelů pomocí předzpracování korpusů.

[8] ADR – `https://en.wikipedia.org/wiki/ADR_(treaty)`

Právě texty pro trénování se podílely hlavní měrou na výsledné výkonosti word2vec modelů.

4.2.1 Metoda bez použití anotací

Během metody je model trénován korpusem obsahujícím texty ze všech digitálně čitelných objednávek v datasetu. Během čtení jsou systematicky zahazovány textové elementy, které nejsou potřeba a mohly by snižovat kvalitu trénování i výsledný model. Mezi takové patří elementy obsahující pouze číselné znaky a elementy obsahující velmi dlouhý text, které je možné s jistotou prohlásit za nepotřebný, takové elementy obsahují přirozený text pojednávající o podmínkách přepravy, pojištění apod. Celkový korpus čítá přes 150 tisíc slov.

Během experimentace byly modely trénovány ve 4 epochách. Jako přesnější se ukázala architektura Skip-gram. Bylo pracováno s rozsahem dimenzí vektorového prostoru mezi 20 až 500, ale tento parametr měl pouze nepatrný efekt na výsledný model.

Takto trénované modely byly použitelné při klasifikaci, ovšem některé skupiny vektorů byly velmi podobné skupinám specifikující opačný jev např. naložení/vyložení. Dalším nedostatkem byl také velký počet irelevantní slov typu křestní jména, názvy firem apod. Tento nedostatek lze do určité míry řešit zmenšením minimálního počtu slov pro zařazení do vektorového prostoru s rizikem, že se do výsledného modelu nedostanou důležitá slova v cizích jazycích nebo netradičně formulovaná slova. Vizualizaci vektorového prostoru je možné vidět na obrázku 4.3

4.2.2 Metoda s použitím anotací

Druhá metoda se snaží řešit výše zmíněné problémy předchozí techniky. Efektivním řešením se ukázalo trénovat word2vec modely předzpracovanými texty. Vytvoření těchto vstupních textů probíhalo přečtením všech anotovaných objednávek a nalezení textových elementů, obsahující hodnoty z anotací. Tyto elementy obsahují informaci, patřící k jednomu z klíčů ze schémy objednávky. U nalezených textových elementů jsou vyhledány oba sousední elementy stejným mechanismem jako při klasickém zpracování objednávky. Na začátek i konec textu sousedních prvků a aktuálního prvku jsou vloženy identifikátory klíče nalezené hodnoty. Takovéto rámce jsou ukládány do souboru, který bude po zpracování všech objednávek sloužit jako trénovací korpus pro word2vec modely. Identifikátory klíčů jsou generovány a je možné je najít ve výsledném slovníku jako iniciály klíče obalené do dvou podtržítek.

Nevýhodou této techniky je pokrytí pouze anotovaných objednávek. Je tedy možné, že některé nestandardní slova se do korpusu nedostanou a chybějící informace tak mohou snížit kvalitu výsledných word2vec modelů. Tento nedostatek ovšem postihuje pouze minimální množství především cizojazyčných objednávek a bude odstraněn během první fáze testování anotováním více objednávek. S menším množstvím dat je také trénování více nestabilní a může vést k poměrně velkým odchylkám mezi jednotlivými procesy učení. Na druhou stranu u slov, které se v korpusu vyskytují máme jistotu, že se nachází v blízkosti informací, které hledáme a jsou pro samotnou klasifikaci důležité. Odstraněním velké části irelevantní slov došlo k výraznému zmenšení trénovacího korpusu a word2vec modely je možné trénovat s mnohem menší velikostí vektorů. Tato vlastnost se pak zásadně promítá i do klasifikačních modelů, u kterých je díky menším vektorům výrazně snížena komplexita. Hlavní výhodou této metody je ale lepší rozdělení vektorového prostoru. U první metody dochází k jevu, kde podobné kategorie např. datum nakládky a vykládky se nachází v dokumentu za sebou, a proto i jejich vektorové reprezentace mají podobný směr. Ve druhé metodě jsou ovšem texty jedné kategorie v trénovacím korpusu seskládány u sebe, a navíc

odděleny identifikátorem klíče dané kategorie. Tato struktura způsobí, že jednotlivé kategorie jsou poté ve vektorovém prostoru mnohem výrazněji odděleny od ostatních kategorií, jak lze vidět na obrázku 4.4.

Experimentací byly nalezeny optimální parametry modelu, jako nejlépe výkonný se ukázal model s velikostí klouzavého okna 4-8, velikostí vektoru 16-32 a architekturou Skip-gram. Po přizpůsobení klasifikačních modelů na nový word2vec model, bylo možné dosáhnout lepších výsledků než s předchozí metodou. Úspěšnost klasifikace byla zvýšena o jednotky procent, byla zvýšena také rychlost trénování i samotné klasifikace.

4.2.3 Textové datasety

Pro trénování klasifikačních modelů se vytváří textové datasety zpracovaných objednávek. Textové datasety mají csv formát, hodnoty jsou oddělovány čárkou. Tyto datasety jsou ukládány podle kategorie a podle toho, jestli zahrnují objednávky z trénovacího nebo testovacího datasetu. Každý řádek reprezentuje jeden kandidátní prvek, obsahuje tedy jeho pozici, texty sousedních prvků a text prvku ve kterém se nachází kandidát (bez samotné hodnoty). Vedle těchto informací důležitých pro samotnou predikci je zde i třída pro trénování modelu, binární číslice určující, jestli kandidát spadá do dané kategorie nebo ne. Dále se na každém řádku nachází cesta k objednávce, ve které se kandidát nachází, tato informace je pouze pro účely ladění.

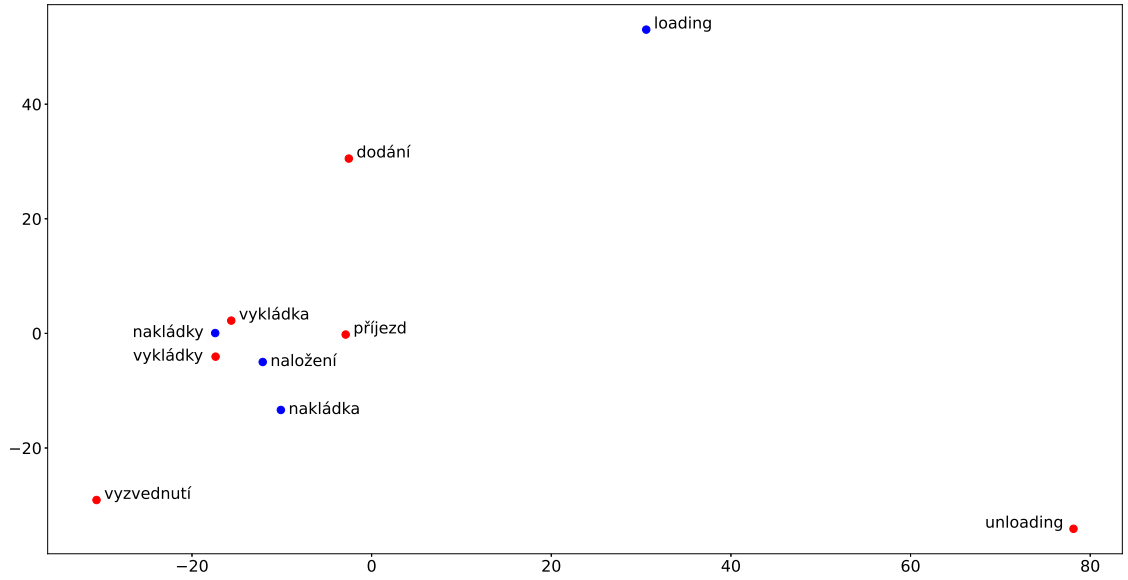
4.3 Klasifikační modely

Klasifikační modely jsou poslední a také nejdůležitější částí rozpoznávání. Kromě počáteční fáze pro rychlé ověření konceptu a sestavení úplné pipeliny produktu byly všechny modely postaveny na neuronových sítích. Tyto sítě i jejich architektury se vyvíjeli spolu s vývojem ostatních částí modulu rozpoznávání.

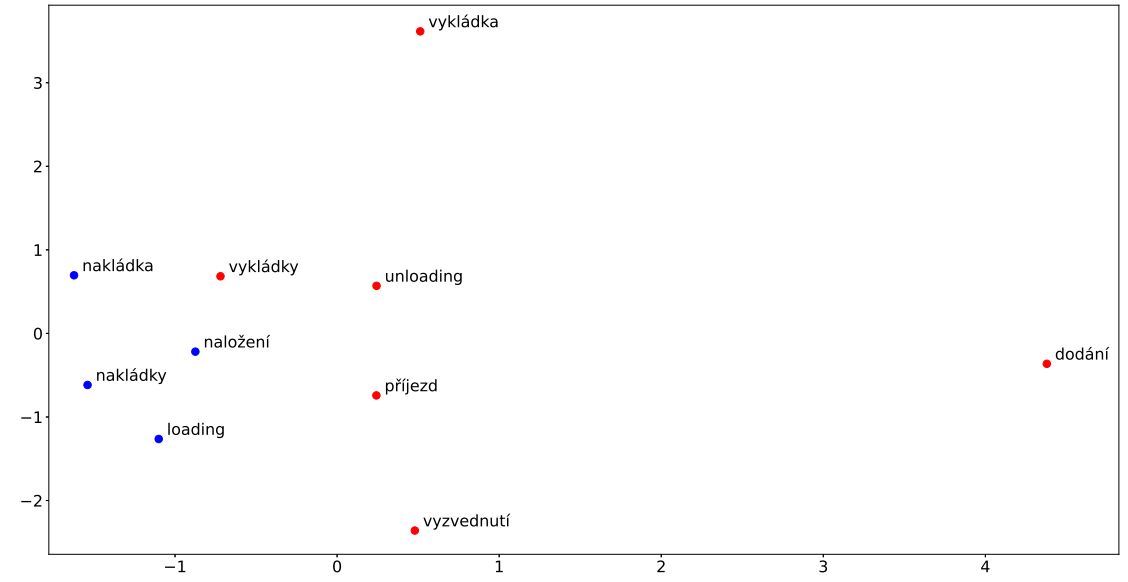
4.3.1 Vícetřídní klasifikační modely

Multiclass neboli vícetřídní klasifikační modely kategorizují vstupní data do jedné z několika kategorií ze schémy objednávek nebo do speciální kategorie, která specifikuje, že daný vzorek nepatří do žádné z kategorií. Výhodou těchto modelů je snížená režie pro práci s modely a časová náročnost výpočtu, jelikož veškerá klasifikace může být prováděna pomocí jediného modelu. Nevýhodou a také důvodem proč tento typ modelu není použit ve finálním produktu je fakt, že pro efektivní vytrénování je potřeba více dat než u binárních modelů. Složitější trénování způsobilo menší úspěšnost, převážně u hůře rozeznatelných kategorií jako jsou datumy. Určité korekce bylo dosaženo manuálním zvýšením váhy pro obtížněji trénovatelné kategorie, ani to ale nestačilo pro potřebnou efektivitu modelu. Bylo také prováděno vyvážení datasetu podle různých metod, také bez většího úspěchu, jejich úspěšnost je možné nalézt v tabulce 4.1.

Architektura Model je konvoluční neuronová síť, mající vstup 3 sekvence vektorů (5x32), ty jsou zpracovány 1D konvoluční vrstvou s 12 filtry a velikosti kernelu 3. Výstupy konvoluční vrstvy jsou zpracovány global max pooling vrstvou (klasická max pooling vrstva s pool velikostí rovnající se velikosti vstupu). Výstupy z pooling vrstev jsou spojeny se vstupem obsahující pozici klasifikovaného prvku. Konec sítě tvoří jedna plně propojená vrstva s 32 neurony, propojená na výstup sítě mající 11 neuronů. U poslední vrstvy byla použita



Obrázek 4.3: Vektorový prostor vytvořený metodou bez anotací a převedeného do 2D pomocí PCA. Jednotlivé body reprezentují vektory resp. slova důležitá pro správnou klasifikaci. Modré body identifikují kategorii související s nakládkou a červené body souvisejí s vykládkou. Lze vidět, že vektorový prostor není ideálně rozdělen a některé fráze mohou být poměrně složité pro klasifikaci.



Obrázek 4.4: Vektorový prostor vytvořený metodou s anotacemi a převedeného do 2D pomocí PCA. Nyní jsou body jednotlivé kategorie seskupeny více u sebe. Klasifikace i trénování klasifikačních modelů tak bude mnohem snazší a přesnější. K podobnému jevu došlo i u ostatních antonymních kategorií např. odběratel–dodavatel.

	datum			zákazník			dopravce			
vyvažovací metoda	objed.	nakl.	vykl.	jméno	ič	dič	jméno	ič	dič	cena
žádná	22.22	22.22	25.00	41.67	36.11	52.78	55.56	52.78	**69.44**	41.67
podvzorkování	22.22	**36.11**	36.11	**61.11**	**80.56**	**77.78**	47.22	61.11	47.22	5.56
vyvážení tříd	**27.78**	19.44	30.56	47.22	61.11	72.22	50.00	**80.56**	**69.44**	**58.33**
SMOTE	25.00	**36.11**	38.89	41.67	**80.56**	69.44	**66.67**	52.78	50.00	13.89
ADASYN	22.22	33.33	**41.67**	33.33	75.00	63.89	55.56	58.33	47.22	19.44

Tabulka 4.1: Výkonnost vícetřídních modelů na testovacích (neviděných) datech. Byly vybrány pouze kategorie, jejichž klasifikace je přímo závislá na klasifikačních modelech. Tučně jsou zvýrazněny nejlepší výsledky. Nejlépe si vedl model trénovaný datasetem s vyvážením tříd, který dosáhl průměrné úspěšnosti 51.66%. Nejhůře dopadl model bez vyvažování s průměrnou úspěšností 41.9%. Modely měly největší problém klasifikovat datumy, ale ani ostatní kategorie neměly velkou úspěšnost.
Málo dat způsobilo nedostatečné, a hlavně nestabilní učení modelů a výsledné modely nejsou použitelné.

aktivační funkce *sigmoid*, jinak byla využita funkce *tanh*. Model byl trénován v 50 epochách s learning rate 0.02, optimalizačním algoritmem Adam a velikostí dávky 6.

4.3.2 Binární klasifikační modely

Druhým typem modelů jsou binární klasifikátory. Pro každou rozpoznávanou kategorii je vytrénován zvláštní model s použitím metody One-vs-All v rámci kategorie, tedy pokud klasifikujeme datum, všechny negativní vzorky budou pocházet také z kandidátů obsahující datum. Tímto rozdělením se u některých kategorií do určité míry minimalizovala nevyváženost. Nejméně vyvážená zůstala kategorie ceny, kde pozitivní vzorky tvořily 15%. Jednotlivé úspěšnosti lze vidět v tabulce 4.2.

Architektura je velice podobná jako u vícetřídních modelů, pouze plně propojená vrstva obsahuje 16 neuronů a je propojena s 1 neuronem určující pravděpodobnost klasifikace. Modely byly trénovány v 50 epochách s mírou učení 0.0015, momentem 0.01 a optimalizátorem stochastického gradientního sestupu s velikostí dávky 2.

4.4 Proces anotace

Anotace jsou soubory ve formátu json obsahující informace jedné objednávky. Za anotaci se považují json soubory pojmenované stejně jako pdf objednávka ve stejné složce. Anotace objednávek probíhá ručně a jsou zadávány pouze informace, které se vyskytuji v souboru a není potřeba je nijak odvozovat. Ruční vytvoření anotace je časově náročný úkon a v budoucnu by měly být anotace vytvářeny hlavně z dat, které zadají samotní uživatelé aplikace. I v takovém případě je ale potřeba informace kontrolovat. Stává se, že především časové údaje se mohou změnit po emailové/telefonické domluvě. V takovém případě bude nutné zjistit, jak často se takové změny dějí a v případě ponechání chybných informací bude potřeba zjistit vliv na trénování klasifikátoru. JSON schéma anotace se nachází v sekci B.

	datum			zákazník			dopravce			
vyvažovací metoda	objed.	nakl.	vykl.	jméno	ič	dič	jméno	ič	dič	cena
žádná	**83.33**	44.44	55.56	**61.11**	**86.11**	**88.89**	**80.56**	**72.22**	**75.00**	63.89
podvzorkování	77.78	36.11	41.60	**61.11**	**86.11**	83.33	72.22	**72.22**	**75.00**	61.11
vyvážení tříd	**83.33**	44.44	44.38	**61.11**	80.56	86.11	**80.56**	**72.22**	72.22	**69.44**
SMOTE	77.78	50.00	63.89	**61.11**	**86.11**	**88.89**	77.78	**72.22**	**75.00**	**69.44**
ADASYN	77.78	**58.33**	**66.67**	**61.11**	**86.11**	**88.89**	**80.56**	**72.22**	**75.00**	**69.44**

Tabulka 4.2: Výkonnost binárních modelů na testovacích (neviděných) datech. Z výsledků je patrné, že binární modely dokáží klasifikovat mnohem lépe než vícetřídní modely zmíněné výše. Je také možné vidět, že vyvážení datasetů již nemá takovou váhu a modely i bez vyvážení dosahují maximálních hodnot. Slabým místem systému stává hledání sousedních polí a nedostatek relevantních informací jdoucí jako vstup do modelů. Modely byly vytrénovány pomocí křížové validace bez hledání ideálních parametrů. U některých modelů nebylo dosaženo ideálních výsledků kvůli stochastičnosti trénování. Nalezené ideální modely jsou popsány v kapitole 7.

4.5 Proces testování

Pro otestování celého rozpoznávacího modulu byl navržen skript, který zpracuje objednávky a výsledek porovná s anotací testované objednávky. Testují se všechny objednávky v určené cestě obsahující anotace. Testovací skript poskytuje možnost vynechat objednávky, které je nutné zpracovat pomocí OCR pro urychlení testování. Anotace každé objednávky obsahuje všechny informace nacházející se přímo v dokumentu a musí být umístěna ve stejné cestě ve formátu json. U všech testovaných objednávek je vypsána úspěšnost rozpoznání, tyto výsledky jsou ukládány a na konci testování je vypsána průměrná úspěšnost rozpoznání u všech hledaných kategorií.

4.5.1 Metody porovnávání

V rámci systému se rozpoznává několik typů dat. Od řetězců jako je adresa nebo název společnosti přes číselné hodnoty až po kategorická data jako je jednotka váhy nebo rozměrů. Podle typu rozpoznávané kategorie je tedy vybrána porovnávací metoda, která jednotlivé kategorie v objednávce ohodnotí. Ohodnocení probíhá na škále od 0 do 100, čím vyšší tím lepší. Z hodnocení jednotlivých kategorií se poté počítá celkové skóre pro každou objednávku, stejně jako přesnost klasifikace jednotlivých kategorií v rámci celého testovacího datasetu. V následujících sekcích bude popsáno, jak jsou jednotlivé kategorie testovány a jak jsou hodnoceny v rámci výkonnosti modelu.

Přesné porovnání je metoda, která přesnou shodu ohodnotí 100 body, jakákoli odchylka od anotace je hodnocena 0 body. Tato metoda se používá pro většinu kategorií: datumy, IČO, DIČ, kvantita zboží, rozměry, váha i jednotky rozměrů a váhy.

Porovnání řetězců je metoda používaná pro porovnání adres a názvu společností. Pro vyhodnocení je použita metrika, která spočítá počet správně určených slov a vydělí je počtem slov v anotaci. Ohodnocení pak vzniká vynásobením tohoto výsledku hodnotou 100.

Porovnání cen je speciální v tom, že správná celková cena může nabývat dvou hodnot, a to buď s DPH nebo bez ní. Pokud je rozpoznaná hodnota pro cenu přesně rovna jedné z těchto cen, je ohodnocena 100 body jinak je pole ohodnocenu 0 body.

Kapitola 5

Návrh webové aplikace

Webová aplikace bude uživatelům sloužit pro snazší vkládání objednávek do systému Cargotic. Pomoci by měla nejen při vyplňování informací, ale také následné kontrole správnosti informací. Webovou aplikaci budou ve velké části využívat technicky méně zdatní uživatelé, a proto je nutné aby ovladatelnost i uživatelské rozhraní bylo co nejvíce uživatelsky přívětivé.

Následující část práce popisuje webovou aplikaci, která slouží především pro demonstrační a testovací účely. Produkt používaný dispečery bude integrace této webové aplikace do platformy Cargotic. Pro ukládání zpracovaných objednávek i jejich dokumentů bude sloužit API platformy Cargotic, dále se tedy budou popisovat pouze klíčové komponenty webové aplikace důležité pro pozdější integraci. Z popisu bude vynecháno ukládání objednávek, schéma a práce s databází i bezpečnost. Implementace do platformy Cargotic je rozpracovaná a po otestování v testovacím prostředí bude dále poskytována širšímu okruhu zákazníků.

Díky demonstrační aplikaci bylo možné vyladit uživatelské rozhraní a ovládání aplikace s lidmi z oboru. Webová aplikace byla tvořena tak, aby bylo možné jednotlivé komponenty lehce znovu využít a integrovat do systému Cargotic. Serverová část aplikace bude po integraci běžet jako samotná služba a bude využívat ostatních služeb platformy. Koncové body (endpointy) API lze vidět v tabulce 5.1.

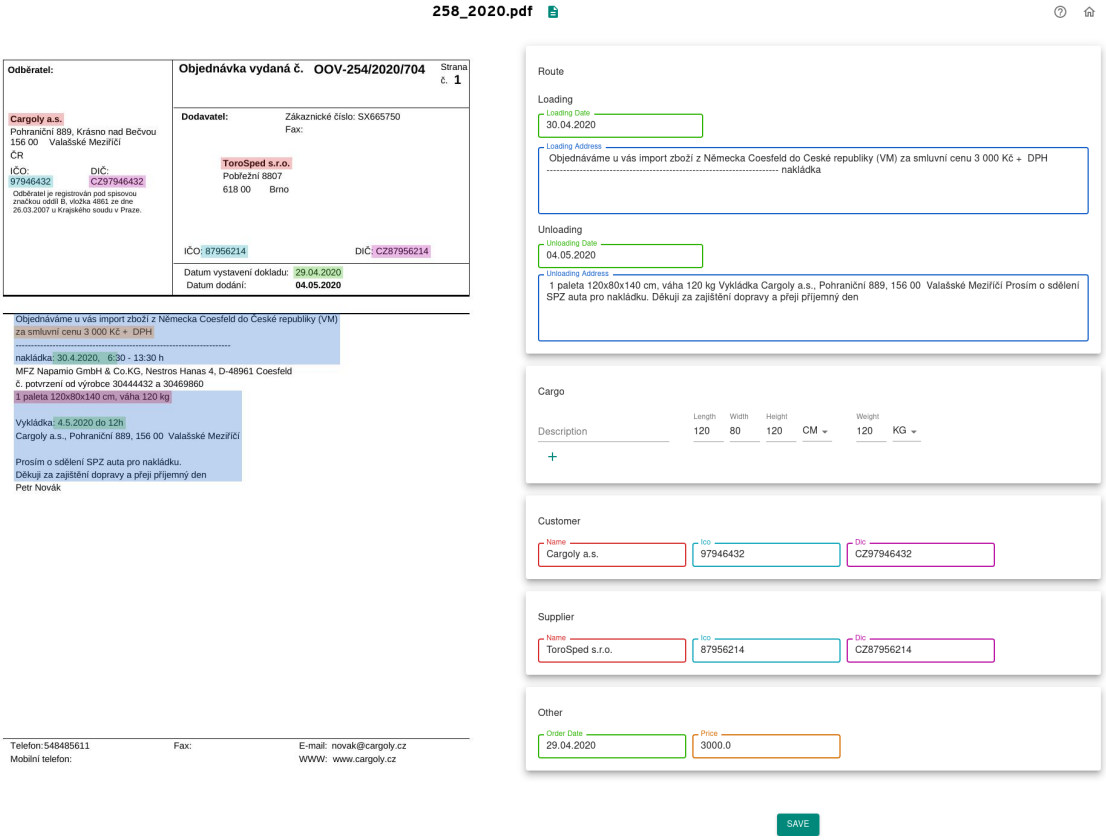
5.1 Analýza požadavků

Webovou aplikaci budou využívat především technicky méně zdatní uživatelé, kteří zvládají ovládání počítačů na kancelářské úrovni. Je tedy nutné vytvořit takovou aplikaci, která bude dostatečně intuitivní a přehledná. Hlavním cílem webové aplikace je ušetření času uživatelům, aplikace by tedy měla umožnit přímou manipulaci s daty a poskytnout potřebné informace na jednom místě. Aplikace by se měla soustředit více na efektivnost než efektnost, neměla by tedy obsahovat animace ani zbytečné dekorativní prvky, které by mohly zpomalit nebo odvést pozornost uživatele. Aplikace by ale měla mít moderní a čistý design a dodržovat UI[1] a UX[2] zásady.

[1] UI – – User interface – Uživatelské Rozhraní
[2] UX – User Experience – Uživatelská zkušenost

HTTP metoda	URL	Popis
POST	/recognition	Přijímá objednávku ve formátu pdf. Vrací zpracovanou objednávku ve formátu JSON.
GET	/images/<image_id>	Vrací výstřižek z objednávky.

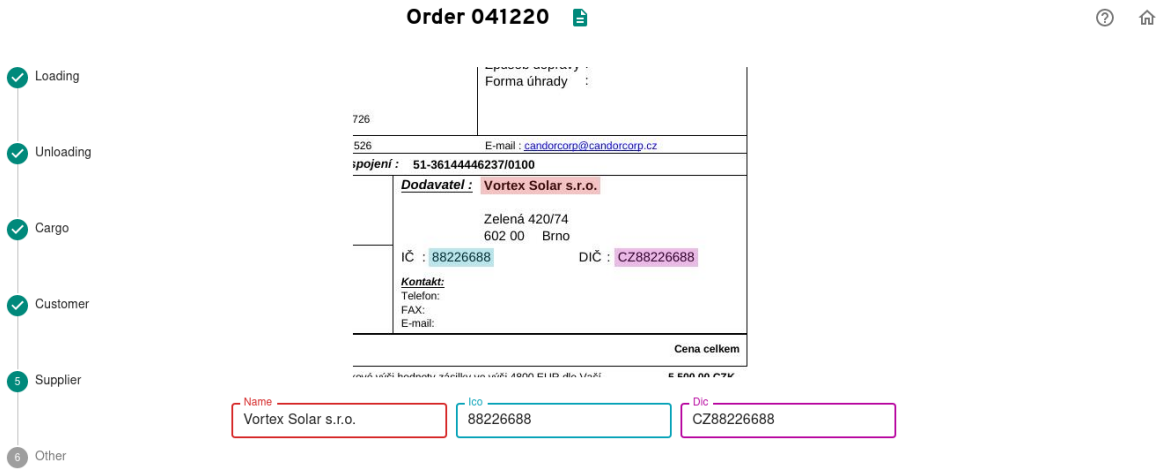
Tabulka 5.1: Návrh API.



Obrázek 5.1: První verze uživatelského rozhraní. Webová aplikace je rozdělena na 2 části. Levá strana obrazovky obsahuje objednávku s vyznačenými hodnotami a na pravé straně jsou formuláře do kterých lze doplnit či upravit data. Ve webové aplikaci z důvodu čitelnosti není vidět celá obrazovka najednou, ale je potřeba scrollovat.

5.2 První návrh

První návrh lze vidět na obrázku 5.1. Návrh vychází z dosavadních aplikací, které se zabývají zpracováním dokumentů viz kapitola 2.2.4. Při testování prvního návrhu bylo ovšem zjištěno, že uživatel je příliš často nucen scrollovat, aby našel potřebné informace v objednávce. U některých objednávek bylo také složité se v barevně vyznačených hodnotách vyznat. V případě, že vyznačování hodnot bylo prováděno podle kategorií, vyskytovalo se v objednávce několik polí se stejnou barvou, což vedlo k častému zmatení uživatele. V případě použití jiného barevného vyznačení pro každé pole bylo naopak v objednávce několik polí s podobným odstínem a nebylo tak snadné k jednotlivým polím v objednávce přiřadit dané pole ve formuláři.



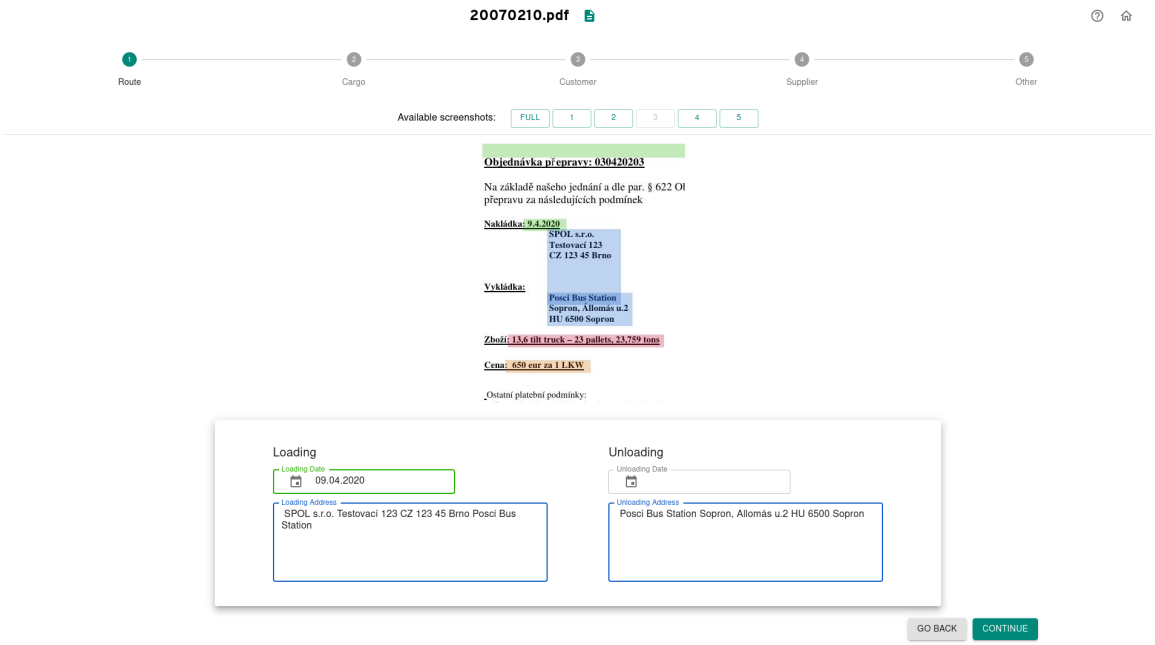
Obrázek 5.2: Druhá verze uživatelského rozhraní. Aplikace obsahuje interaktivní ukazatel průběhu kontroly na levé straně obrazovky. Ve středu obrazovky je poté zobrazen výstřižek sekce a formulář k dané sekci.

5.3 Druhý návrh

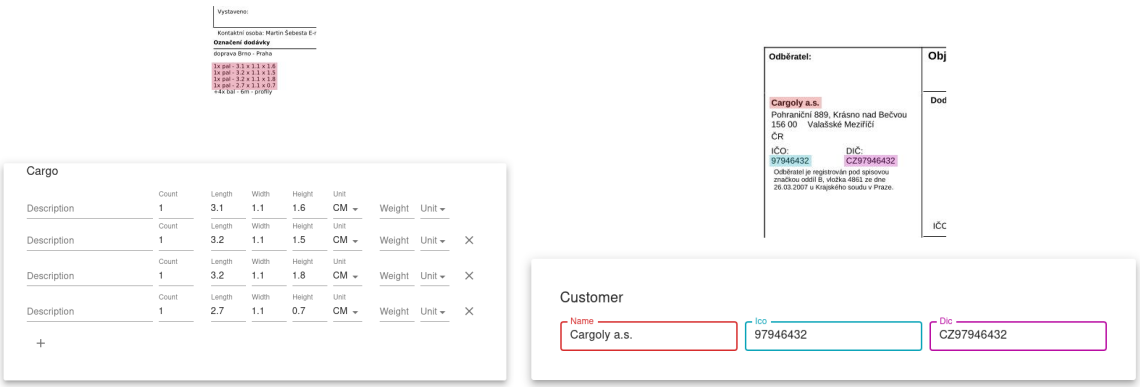
U druhého návrhu bylo využito znalosti typu dokumentu. Pokud všechny dokumenty nehledě na šablonu obsahují stejné informace, pak je možné objednávku rozdělit do několika sekcí. Každá sekce obsahuje informace určitého typu. Pro všechny sekce jsou pak vytvořeny výstřižky zobrazující pouze danou část objednávky. Uživatel pak nekontroluje celou objednávku najednou, ale může si dovolit postupovat po krocích, což vede k výraznému zlepšení přehlednosti a tím i rychlosti kontroly. V návrhu se vyskytují sekce nakládky, vykládky, nákladu, zákazníka, dodavatele a ostatní. Pro každou sekci byl také vytvořen formulář umožňující upravovat jednotlivé informace v sekci. Návrh lze vidět na obrázku 5.2.

5.4 Finální návrh

Po testování s budoucím uživatelem nástroje bylo zjištěno, že není jasné zda při kliknutí na ukazatel jsou uloženy vyplněné informace, bylo tedy přidáno tlačítko pro pokračování, které uživatele automaticky přesměruje na další krok kontroly. Bylo také zjištěno, že výstřižky sekcí nakládka a vykládka jsou často překrývány a jelikož obsahují málo informací je možné je spojit do jedné sekce. Bylo přidáno další usnadnění v podobě možnosti zobrazení i jiných výstřižků, než které patří k sekci, což umožní uživateli plynule doplnit informace, které systém nenašel a které tedy nejsou zahrnuty do výstřižku. S touto změnou se také přesunul ukazatel průběhu pro zachování konzistentnosti uživatelského rozhraní. Některé formuláře lze vidět na obrázku 5.4. Celkový návrh lze vidět na obrázku 5.3.



Obrázek 5.3: Finální verze uživatelského rozhraní pro ověření rozpoznávání je rozdělena na 3 části. Ve vrchní části se nachází ukazatel průběhu kontroly a tlačítka pro zobrazení nalezených výstřižků. Pod vrchní částí se nachází nalezený výstřižek pro zrovna kontrolovanou sekci objednávky. V případě že pro daný segment nebyl žádný výstřižek nalezen je uživatel na tuto skutečnost upozorněn a může si zobrazit celou objednávkou nebo některý z ostatních výstřižků. Ve spodní části obrazovky je pak formulář pro kontrolu a případnou změnu či doplnění informací k dané sekci.



Obrázek 5.4: Formuláře webové aplikace. Na levé straně obrázku lze vidět formulář pro úpravu nákladu. Na pravé straně se nachází formulář pro úpravu subjektů, tedy zákazník a dopravce.

Kapitola 6

Implementace

Projekt torv[1] byl rozdělen do tří částí.

6.1 Datové složky

obsahují složky, které jsou potřeba pro práci samotného modulu rozpoznávání tak i serveru.

models obsahuje vytrénované klasifikační modely.

data obsahuje pomocné soubory, který modul využívá. Jedná se o schému objednávky a seznam měst.

w2v obsahuje word2vec modely a textové korpusy používané pro trénování word2vec modelů.

6.2 Rozpoznávací knihovna

Rozpoznávací knihovna v Pythonu je jádro této aplikace, obsahuje skripty pro zpracování a rozpoznání objednávek, které budou využívány serverem, ale mohou být používány i z terminálu nebo být napojeny na jiné aplikace. Dále se zde nachází skripty pro vytváření textových datasetů, word2vec modelů a klasifikačních modelů. Pro usnadnění dalšího vývoje, jsou zde k dispozici také skript na vyhodnocení modelů, vizualizaci vektorového prostoru word2vec modelů a testovací skript pro vyhodnocení úspěšnosti nástroje. Modul byl napsán v jazyce Python[2] s výjimkou hledání měst, které bylo z důvodu rychlosti napsáno v jazyce C[3].

torv

obsahuje všechny potřebné zdrojové kódy pro zprovoznění rozpoznávacího programu.

`addc.c` je pomocný program pro hledání měst v textu. Před použitím je nutné zkompilovat pro používanou architekturu. Jako vstup dostává text a cestu k souboru obsahující města.

[1] torv – Transport Order Recognition and Verification
[2] Python – `https://www.python.org/`
[3] C – `http://www.open-std.org/jtc1/sc22/wg14`

`constants.py` obsahuje globálně používané konstanty.
`main.py` je skript sdružující ostatní funkce pro zpracování objednávky.

datasets obsahuje textové datasety viz 4.2.3. Datasety určené pro trénování mají prefix train, datasety pro testování mají prefix test.

logs obsahuje logy z trénování modelů.

recognition obsahuje skripty pro čtení a zpracování objednávek.

- `pdf_parser.py` obsahuje funkce nutné pro čtení pdf dokumentů, detekci CID fontů a zpracování dokumentů pomocí OCR.
- `order_recognition.py` obsahuje funkce pro samotné rozpoznání a klasifikaci již přečtené objednávky.
- `entity_recognition.py` obsahuje funkce pro identifikaci entit v textových elementech dokumentu, které dále určují klasifikaci elementu.

classification obsahuje skripty pro vytvoření objektů nutných ke klasifikaci – textové datasety, klasifikační a word2vec modely.

- `binary_dataset.py` slouží pro vytvoření textových datasetů. Textové datasety jsou vytvářeny zvlášť pro trénovací a testovací data. Textový datasety jsou určeny pro binární klasifikaci, pro každou kategorii je vytvořeny dva datasety – trénovací a testovací.
- `train_model.py` slouží k trénování binárních klasifikačních neuronových sítí. Obsahuje funkce pro klasické trénování i trénování s k-fold cross validací. Pro trénování je nutné poskytnout již vytrénovaný word2vec model a textové datasety.
- `w2v.py` je skript pro práci s word2vec v rámci projektu. Umožňuje vytvořit čistý textový korpus z objednávek, tedy přepis pdf do prostého textu. Skript také obsahuje funkce pro vytvoření předzpracovaného korpusu, které jsou vytvořeny z anotací jak bylo popsáno v kapitole 4.2.2. Z těchto korpusů je možné vytrénovat word2vec modely s různými parametry.

classification/experimental obsahuje moduly pro klasifikaci, které nejsou používány ve finální verzi aplikace, kvůli malé úspěšnosti viz kapitola 4.3.1. Jejich využití by mohlo nastat s nárůstem trénovacích dat.

- `multiclass_dataset.py` obsahuje funkce pro vytvoření textových datasetů pro vícetřídní klasifikaci. Pro následné použití s vícetřídními klasifikátory je opět zapotřebí pár trénovacího a testovacího datasetu, oba obsahující záznamy ke všem kategoriím.
- `multiclass_train_model.py` obsahuje funkce na vytvoření a trénování vícetřídních klasifikátorů. Pro trénování je opět potřeba word2vec model a vícetřídní textové datasety.

utils obsahuje pomocné funkce pro zpracování objednávek, výstřižků, pro práci s modely, vektory a obecně používané pomocné funkce.

- `image_utils.py` obsahuje funkce pro práci s výstřižky. Konkrétně se jedná o oříznutí výstřižků, zvýraznění nalezených hodnot a ukládání.
- `model_utils.py` obsahuje funkce pro práci s word2vec modely a pomocné funkce pro práci s jejich vektory. Dále skript obsahuje funkce pro vytváření architektury klasifikačních modelů a pro zpracování dat předávaných modelu.
- `order_utils.py` obsahuje funkce pro práci s objednávkami. Jedná se o funkce pro formátování dat, hledání sousedů a správa kontextů potřebných pro vytváření výstřižků.
- `utils.py` obsahuje funkce pro práci s řetězci, slovníky, datumy a jiné funkce používané v celém projektu.
- `w2v_visualization.py` umožňuje vizualizovat word2vec model pomocí PCA. Specifikováním klíčových slov z různých kategorií je možné lépe pochopit a vyhodnotit jednotlivé word2vec modely.
- `eval_models.py` je skript sloužící pro vyhodnocení binárních modelů nad datasetem objednávek. Skript poskytuje hodnoty matice záměn a důležité metriky jako F1 skóre, přesnost, preciznost a senzitivita.

tests

- `eval_test.py` je test který vyhodnotí všechny objednávky ve specifikované složce. Pro urychlení testování je možné vyhodnocovat pouze objednávky, u kterých není potřeba použít OCR.

6.3 Klientská část aplikace

Webová aplikace je napsána v JavaScriptu[4] s využitím knihovny pro tvorbu uživatelského rozhraní, React[5]. Design aplikace byl navržen podle Material Design[6], stejně jako využité komponenty z knihovny Material UI[7]. Technologie byly zvoleny podle používaných technologií v systému Cargotic, pro co možná nejsnazší integraci.

src

obsahuje zdrojové kódy k webové aplikaci.

forms obsahuje formuláře použité pro vyplňování informací.

components obsahuje speciální komponenty jako je textové pole nebo komponenta pro výběr datumu se specifickým rámečkem podle typu hodnoty.

[4] JavaScript – `https://www.javascript.com`
[5] React – `https://reactjs.org`
[6] Material Design – `https://material.io`
[7] Material UI – `https://material-ui.com`

OrderValidation obsahuje stránku pro editaci objednávek.

OrderContext obsahuje kontext objednávky a provider kontextu.

public

obsahuje index aplikace, manifest definující základní prvky webové aplikace a obrázek používaný na úvodní obrazovce.

6.4 Serverová část aplikace

Webový server je napsán v jazyce Python s využitím frameworku Django[8].

`manage.py` je hlavní skript sloužící pro práci se serverem.

orders obsahuje pdf dokumenty objednávek, které byly serverem zpracovány.

screenshots obsahuje vytvořené screenshoty objednávek na které je možné se dotazovat.

server

- `controller.py` definuje logiku jednotlivých endpointů API.
- `settings.py` obsahuje základní nastavení serveru.
- `urls.py` obsahuje seznam endpointů.
- `wsgi.py` obsahuje Web Server Gateway Interface[9].

[8] Django – `https://www.djangoproject.com`
[9] WSGI - `https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi`

Kapitola 7

Hodnocení rozpoznávacího systému

7.1 Dosažená přesnost klasifikace

Porovnání jednotlivých typů modelů proběhlo v kapitole 4.3. Následující sekce se zaměří na úspěšnost klasifikace v celém systému. Udávané hodnoty jsou přesnosti klasifikace provedené nad testovacím datasetem popsaným blíže v kapitole 2.3 a ohodnocené testovacím skriptem. Postup testování je blíže popsán v kapitole 4.5. Histogram celkové úspěšnosti rozpoznávání lze vidět na obrázku 7.1.

7.1.1 Úspěšnost základních kategorií

Základní kategorie jsou určeny pomocí klasifikačních modelů. V tabulce 7.1 můžeme vidět porovnání výsledků na objednávkách při kterých není nutné použít OCR (*torv-clean*) a výsledky na všech objednávkách (*torv*).

7.1.2 Úspěšnost nalezení adres v objednávkách

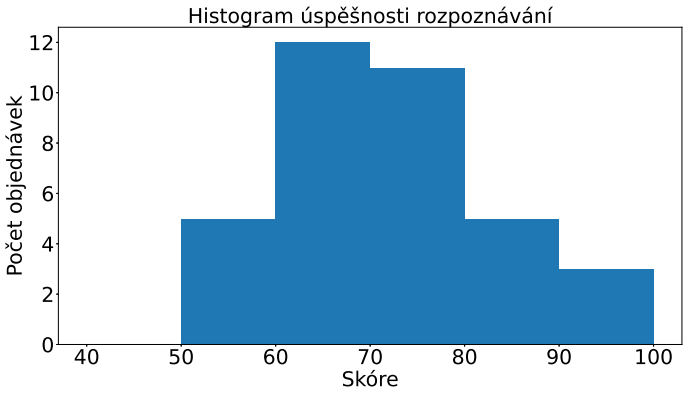
K speciálním kategoriím patří adresy. Ty spoléhají na klasifikačních modely, konkrétně na správné určení datumů, ale mimo to jejich úspěšné rozpoznání závisí také na mechanismu správného nalezení měst v rámci objednávky. Úspěšnost systému při rozpoznání adres lze nalézt v tabulce 7.2.

7.1.3 Úspěšnost nákladu

Množina kategorií specifikující náklad patří zatím ke kategoriím určované čistě podle heuristik, protože anotace neobsahují dostatek dat pro vytrénování efektivních klasifikačních modelů. Dosavadní výsledky lze nalézt v tabulce 7.3.

7.2 Porovnání s konkurencí

V následující sekci budou porovnány výsledky námi navrženého řešení s konkurenčními řešeními. Jak bylo zmíněno v kapitole 2.2.4, porovnání s existujícími nástroji není jednoduché. Komerční nástroje pro obecné zpracování dokumentů neposkytují plnou funkcionalitu bezplatně a vědecké články se zabývají převážně zpracováním faktur a účtenek, nejčastěji s vlastními datasety. V následujících porovnáních se zaměříme převážně na výsledky systémů zpracovávající faktury, které jsou podobnější přepravním dokumentům než účtenky.



Obrázek 7.1: Histogram celkové úspěšnosti rozpoznání. Celkové skóre objednávky je průměr skóre jednotlivých polí jak je popsáno v kapitole4.5.

	datum			zákazník			dopravce			
	objed.	nakl.	vykl.	jméno	ičo	dič	jméno	ičo	dič	cena
torv-clean	0.833	0.633	0.600	0.600	0.866	0.933	0.833	0.700	0.800	0.700
torv	0.861	0.583	0.666	0.611	0.861	0.888	0.805	0.722	0.777	0.694

Tabulka 7.1: Úspěšnost klasifikace základních kategorií. Z výsledků vyplývá, že nutnost zpracovat objednávky pomocí OCR nijak výrazně neovlivní výsledky modelu. Odchylky mezi výsledky jsou především náhodného rázu a budou se zmenšovat spolu se sesbíráním více dat.

Protože některé systémy rozpoznávají i pro nás nerelevantní hodnoty, budou porovnávány pouze stejné nebo podobné kategorie informací.

7.2.1 Majumder et al. 2020

Vědecký článek publikovaný v roce 2020 týmem výzkumníků ze společnosti Google, Majumder et al. [10] se zabývá extrakcí informací z formulářových dokumentů, konkrétně faktur a účtenek, za pomoci modelu založeného na transformátorech a self-attention mechanismu. V rámci práce byl sesbírán a anotován dataset o velikosti skoro 500 účtenek a 15 000 faktur. Autoři se zaměřili na extrakci datumů a celkové ceny u účtenek. U faktur bylo rozpoznáváno celkem 8 polí, kde extrahované hodnoty spadaly do jednoho z typů cena, datum, řetězec. Výsledky práce na fakturách byly dále rozebrány v článku [16]. Porovnání s navrženým řešením lze vidět v tabulce 7.4. Byly porovnány pouze pole, které spolu souvisejí, tedy datum faktury/objednávky, datum dodání/vyložení a celková cena.

7.2.2 Chargrid, Katii et al. 2018

Chargrid je systém sepsaný v práci [6] od Katti el al. Systém měl k dispozici 12 000 faktur od různých výrobců a v různých jazycích. Autoři navrhli několik modelů, které klasifikovali do celkem 8 kategorií. Jako ojedinělý systém, Chargrid extrahuje z dokumentů i adresy, konkrétně tedy adresu výrobce.

	nakládka	vykládka
torv-clean	0.396	0.496
torv	0.406	0.483

Tabulka 7.2: Úspěšnost rozpoznání adres. Úspěšné nalezení adres je nepřímo závislé na úspěšnosti klasifikačních modelů. Lze tedy předpokládat, že s přibývajícími daty a úspěšnějším rozpoznáváním souvisejících datumů, vzroste i úspěšnost nalezení adres.

	kvantita	délka	šířka	výška	jednotka rozměrů	váha	jednotka váhy
torv-clean	0.467	0.763	0.795	0.872	0.861	0.774	0.842
torv	0.472	0.750	0.758	0.875	0.834	0.756	0.844

Tabulka 7.3: Úspěšnost rozpoznání nákladu v objednávkách. Výsledky ukazují že rozpoznávání pomocí heuristik funguje poměrně dobře na rozměry nákladu, váhy i kategorické hodnoty určující jednotky. Menší úspěšnost u kvantity zboží je způsobena především vzácným výskytem v objednávkách, jelikož se jedná o sekundárních informaci a často bývá vynechána nebo nahrazena 0.

V práci je výkonnost systému měřena jejich vlastní metrikou. Pro predikci je spočítán počet nutných úprav všech instancí jedné kategorie pro dosažení ground truth (základní pravda) jednotlivých instancí. Tento počet úprav je poté vydělen celkovou délku instancí této kategorie. Tato metrika C je definována v rovnici 7.1, kde I je počet nutných vložení, M je počet nutných modifikací a D je počet nutných smazání pro dosažení ground truth. Rovnice byla převzata z práce [6].

$$C = 1 - \frac{I + D + M}{N} \tag{7.1}$$

Tato metrika může být vypovídající u systémů pracujících na úrovni řetězců, pokud ale stejnou metriku aplikujeme na náš systém, který klasifikuje entity, nebudou výsledky vypovídající. Příkladem může být špatná klasifikace datumů, pokud náš systém nesprávně klasifikuje datum objednávky „23.05.2021“ jako datum nakládky, které má ground truth „26.05.2021“, bude podle metriky výkonnost modelu 0.9. V takovém případě by náš model dosahoval velmi vysoké úspěšnosti i při špatné klasifikaci všech vzorků. V následujícím porovnání v tabulce 7.5 bude proto pro náš model zachována metrika úspěšnosti klasifikace jak je popsána v sekci 4.5, porovnání je tedy potřeba brát s určitou rezervou.

7.2.3 Krieger et al. 2021

Práce publikovaná Krieger et al. [7] se zabývá extrakcí informací z faktur pomocí grafových neuronových sítí. V rámci práce bylo sesbíráno přes 1100 faktur od více než 300 unikátních výrobců. Modely dokáží klasifikovat informace do kategorie datum, číslo nebo cena faktury. Číslo objednávky není v rámci naší práce relevantní a je proto vynecháno z porovnání, které lze najít v tabulce 7.6.

	invoice date	delivery date	total amount
Majumder et al.	**0.940**	0.667	**0.858**
torv	0.729	**0.730**	0.632

Tabulka 7.4: Porovnání F1 skóre s výsledky z práce navrhující model založený na transformátorech a self-attention mechanismu. Z výsledků lze vidět, že náš model si vedl lépe na datu dodání, které není na všech dokumentech úplně běžné. U informací uváděných na každé faktuře jako je datum faktury nebo cena si vedl lépe porovnávaný model.

	invoice date	invoice amount	vendor name	vendor address	item quantity
sequential	0.839	0.791	0.289	0.169	-0.018
image-only	0.456	0.689	0.196	0.139	0.467
CG-net	0.837	**0.807**	0.360	0.391	**0.652**
CGH-C32	0.804	0.779	0.320	0.314	0.640
CGH-C64	0.842	0.801	0.342	0.368	0.645
torv	**0.861**	0.694	**0.805**	**0.483**	0.472

Tabulka 7.5: Porovnání klasifikace se systémy Chargrid (*CG* – chargrid, *CGH* – chargrid-hybrid). Z výsledků lze vidět, že náš systém si lépe vede na řetězcových datech jako jsou jména nebo adresy subjektů, podobná úspěšnost klasifikace je u datumů a hůře si náš systém vede u čistě číselných dat jako je kvantita nebo cena.

7.2.4 Lie et al. 2019

Práce publikovaná Liu et al. [9] se zabývá návrhem systému pro extrakci informací z faktur za pomoci grafových konvolučních neuronových sítí. Tyto modely byly porovnány se staršími baseline modely BiLSTM-CRF [5]. Výsledky byly prezentovány klasifikací 6 polí na datasetu 3000 oficiálních čínských faktur, které mají fixní šablonu. Porovnání lze najít v tabulce 7.7.

7.3 Rychlost rozpoznávání

Pro urychlení práce uživatelům je nutné aby systém dokázal zpracovávat požadavky v reálném čase. Na testovacích objednávkách byly tedy změřeny časové náročnosti jednotlivých úkonů. Během testování byla zanedbána časová náročnost síťových operací jako posílání požadavků či příjem dat. Testovaní proběhlo na stroji s Intel CPU i5-4210M 3.20 GHz[1] a klasifikační modely využívali pouze toto CPU.

Rychlost celého zpracování je ovlivněna především použitím OCR a jestli se vytváří výstřižky. Zpracování objednávky bez použití OCR a bez vytváření výstřižků trvá v průměru $\tilde{4}80$ ms. V případě použití OCR opět bez výstřižků, trvá zpracování $\tilde{4}430$ ms. Vytváření výstřižků trvá v obou případech $\tilde{2}50$ ms.

[1] `https://ark.intel.com/content/www/us/en/ark/products/81012/intel-core-i5-4210m-processor-3m-cache-up-to-3-20-ghz.html`

	invoice date	total amount
Krieger et al.	**0.905**	**0.820**
torv	0.729	0.632

Tabulka 7.6: Porovnání F1 skóre s přístupem grafových neuronových sítí. Porovnávaný systém byl vytrénován pro klasifikaci pouze 3 polí ve kterých je velmi efektivní a výkonnější než náš systém.

	invoice date	price	buyer name	seller name
Baseline I	0.962	0.527	0.402	0.681
Baseline II	**0.963**	0.910	0.797	0.731
Liu et al.	**0.963**	**0.943**	0.833	**0.782**
torv	0.729	0.632	**0.877**	0.727

Tabulka 7.7: Porovnání F1 skóre porovnatelných polí s přístupem grafových konvolučních neuronových sítí. I když byl porovnávaný model trénován na fixních šablonách, dokáže náš systém podat porovnatelné výsledky při hledání jmen subjektů. Při klasifikaci ostatních kategorií porovnávaný model podal lepší výsledky.

Rychlost klasifikace do kategorií je kolem 200 ms. V tomto časovém údaji není zahrnuta klasifikace adres.

Rychlost klasifikace adres se pohybuje kolem 110 ms. Dostačující rychlosti bylo dosaženo použitím jazyka C pro vyhledání měst. V Pythonu stejná operace trvala kolem 3 s.

7.4 Celkové zhodnocení

Při vyhodnocení a porovnání navrženého systému s podobnými pracemi jsme dospěli k několika závěrům.

- Slabinou našeho systému je rozpoznání obecných informací jako cena nebo datum faktury, kde úspěšnost klasifikace zaostává před ostatními systémy.
- Náš systém dokáže úspěšněji rozpoznat neobvyklé informace jako jsou adresy nebo data doručení, nadprůměrné výsledky má také v oblasti rozpoznání řetězcových dat jako jsou názvy společností.
- Zatímco náš systém byl vytrénován na 105 dokumentech, ostatní systémy měli k dispozici často tisíce až deseti-tisíce dokumentů.
- Ostatní systémy často dokáží rozpoznat pouze několik málo polí, které by samy o sobě neměly v řešení našeho problému takový význam a rychlost práce by ušetřily pouze minimálně.
- Navržený systém je dostatečně rychlý pro použití v reálném čase.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit návrh a implementaci webového nástroje, který usnadní práci dispečerům spedičních a dopravních firem pomocí automatizovaného rozpoznání důležitých informací v objednávkách. Díky rozpoznání nemusí být všechny informace ručně přepisovány dispečery, což vede k ušetření času.

V rámci práce byl shromážděn a anotován dataset postačující k vytvoření efektivního systému. Před začátkem jeho vývoje byla prostudována problematika extrakce informací z dokumentů a ze získaných znalostí byl navrhnut systém pro rozpoznání důležitých informací v přepravních dokumentech. Návrh byl poté implementován a různými experimenty postupně zlepšován. Pro co nejefektivnější využití vyvinutého nástroje bylo navrhnuto uživatelské rozhraní. Následoval vývoj webové aplikace včetně serverové části, která slouží nejen pro zpracování objednávek, ale i pro sběr dalších dat, pomocí kterých může být systém dále zlepšován.

Výsledkem je funkční nástroj, který dokáže v reálném čase zpracovávat přepravní dokumenty. Úspěšnost rozpoznání byla porovnána s pracemi zabývající se podobnými problémy a v několika ohledech je dokázala i překonat. Na druhé straně systém není dokonalý a v některých kategoriích nepodává požadované výsledky. K rozpoznávacímu nástroji byla vyvinuta webová aplikace a po konzultacích s budoucími uživateli byla postupně vylepšována. Ukázku použití aplikace lze najít na videu[1].

Implementace do administračního systému Cargotic aktuálně probíhá a po důkladném otestování nástroje v testovacím prostředí bude poskytnut všem jeho uživatelům. Ve vývoji jak rozpoznávacího systému, tak i webové aplikace bude nadále pokračováno. V rámci rozpoznávání bude s přibývajícími daty zajímavé pozorovat vývoj modelů a spolu s další experimentací by mělo dojít k dalším zlepšením v úspěšnosti rozpoznávání. Následujícím rozšířením v rozpoznávacím modulu bude přidání dalších polí k rozpoznání, které dále ulehčí práci dispečerům. V rámci webové aplikace by bylo zajímavým rozšířením kopírování textu přímo z objednávky do formulářů pro pole, kde rozpoznání nebylo úspěšné.

[1] `https://www.youtube.com/watch?v=R-kLflXRjeM`

Literatura

[1] Alom, M. Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P. et al. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*. Březen 2019, sv. 8, č. 3, s. 292. DOI: 10.3390/electronics8030292.

[2] Chiticariu, L., Li, Y. a Reiss, F. R. Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, říjen 2013, s. 827–832. Dostupné z: `https://www.aclweb.org/anthology/D13-1079`.

[3] Denk, T. I. a Reisswig, C. *BERTgrid: Contextualized Embedding for 2D Document Representation and Understanding*. 2019.

[4] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Huang, Z., Xu, W. a Yu, K. *Bidirectional LSTM-CRF Models for Sequence Tagging*. 2015.

[6] Katti, A. R., Reisswig, C., Guder, C., Brarda, S., Bickel, S. et al. *Chargrid: Towards Understanding 2D Documents*. 2018.

[7] Krieger, F., Drews, P., Funk, B. a Wobbe, T. Information Extraction from Invoices: A Graph Neural Network Approach for Datasets with High Layout Variety. In:. Březen 2021.

[8] Li, F.-F., Krishna, R. a Xu, D. *CS231n Convolutional Neural Networks for Visual Recognition*. 2021. Dostupné z: `https://cs231n.github.io/convolutional-networks`.

[9] Liu, X., Gao, F., Zhang, Q. a Zhao, H. *Graph Convolution for Multimodal Information Extraction from Visually Rich Documents*. 2019.

[10] Majumder, B. P., Potti, N., Tata, S., Wendt, J. B., Zhao, Q. et al. Representation Learning for Information Extraction from Form-like Documents. *Association for Computational Linguistics*. 2020.

[11] Mikolov, T., Chen, K., Corrado, G. a Dean, J. *Efficient Estimation of Word Representations in Vector Space*. 2013.

[12] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. a Dean, J. *Distributed Representations of Words and Phrases and their Compositionality*. 2013.

[13] More, A. *Survey of resampling techniques for improving classification performance in unbalanced datasets.* 2016.

[14] Palm, R. B., Winther, O. a Laws, F. *CloudScan - A configuration-free invoice analysis system using recurrent neural networks.* 2017.

[15] Pascanu, R., Mikolov, T. a Bengio, Y. *On the difficulty of training Recurrent Neural Networks.* 2012.

[16] Tata, S. *Extracting Structured Data from Templatic Documents.* 2020. Dostupné z: https://ai.googleblog.com/2020/06/extracting-structured-data-from.html.

[17] Theodoridis, S. a Koutroumbas, K. *Pattern Recognition, 4th Edition.* 4. vyd. Academic Press, 2009.

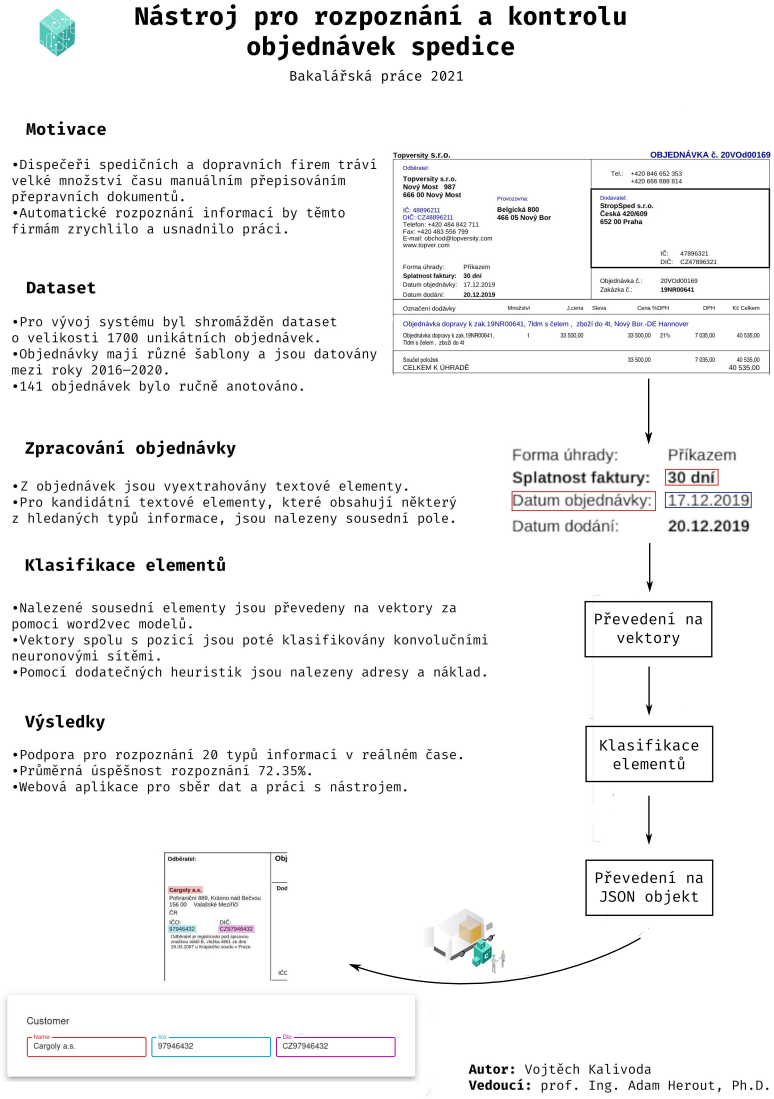
[18] Yin, W., Kann, K., Yu, M. a Schütze, H. *Comparative Study of CNN and RNN for Natural Language Processing.* 2017.

[19] Zhang, A., Lipton, Z. C., Li, M. a Smola, A. J. *Dive into Deep Learning.* 2020. https://d2l.ai.

[20] Zhao, X., Niu, E., Wu, Z. a Wang, X. *CUTIE: Learning to Understand Documents with Convolutional Universal Text Information Extractor.* 2019.

Příloha A

Plakát



Obrázek A.1: Plakát.

Příloha B

JSON schéma anotace

```
1
2  "$schema": "http://json-schema.org/draft-07/schema",
3  "type": "object",
4  "properties": {
5      "shipment": {
6          "type": "object",
7          "properties": {
8              "orderDate": { "type": "string" },
9              "customer": {
10                 "type": "object",
11                 "properties": {
12                     "name": { "type": "string" },
13                     "ico": { "type": "string" },
14                     "dic": { "type": "string" }
15                 }
16             },
17             "supplier": {
18                 "type": "object",
19                 "properties": {
20                     "name": { "type": "string" },
21                     "ico": { "type": "string" },
22                     "dic": { "type": "string" }
23                 }
24             },
25             "price": { "type": "number" },
26             "currency": { "type": "string" }
27         }
28     },
29     "journey": {
30         "type": "object",
31         "properties": {
32             "loading": {
33                 "type": "object",
34                 "properties": {
35                     "date": { "type": "string" },
```

```
                    "address": { "type": "string" }
                }
            },
            "unloading": {
                "type": "object",
                "properties": {
                    "date": { "type": "string" },
                    "address": { "type": "string" }
                }
            }
        }
    },
    "cargo": {
        "type": "array",
        "items": {
            "anyOf": [
                {
                    "type": "object",
                    "properties": {
                        "description": { "type": "string" },
                        "count": { "type": "string" },
                        "weight": { "type": "number" },
                        "weightUnit": { "type": "string" },
                        "length": { "type": "number" },
                        "width": { "type": "number" },
                        "height": { "type": "number" },
                        "lengthUnit": { "type": "string" }
                    }
                }
            ]
        }
    }
}

```